Read Me

First click on the .exe file named ‘Installation’

This will then create all the empty files

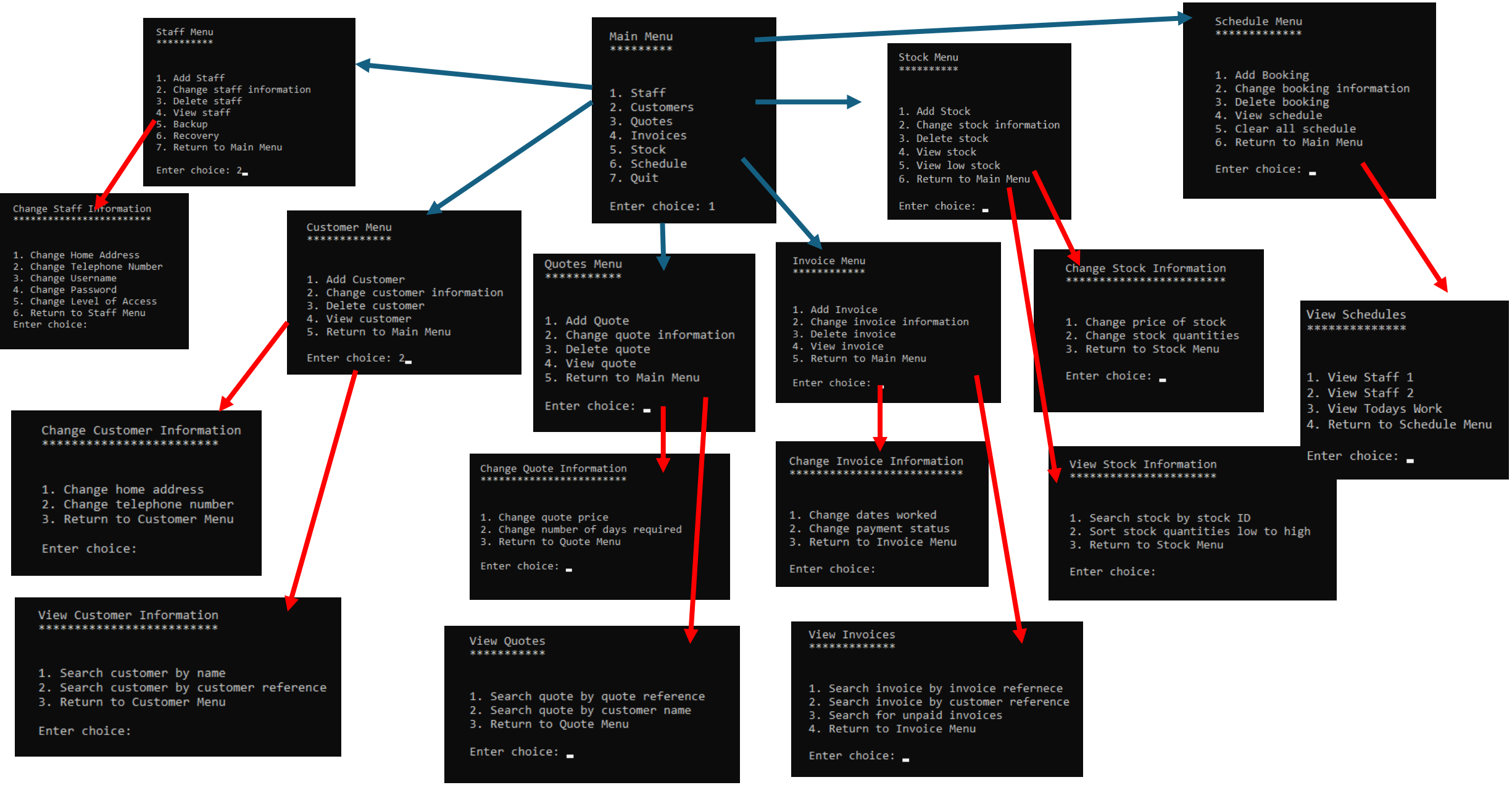
Next click the .exe file called ‘SoftwareDev’

This will then allow you to enter the details for the first staff member, note that the access level will already be assigned as the highest access level – 3.

Once the information is added it will then ask you to enter the username and password you have just entered this will then grant you access to the system.

When adding a quote ensure there is stock added for working out the cost of materials.

Documenting the UI


Main Menu
****
1. Staff
2. Customers
3. Quotes
4. Invoices
5. Stock
6. Schedule
7. Quit
Enter choice: 1
Staff Menu
****
1. Add Staff
2. Change staff information
3. Delete staff
4. View staff
5. Backup
6. Recovery
7. Return to Main Menu
Enter choice: 2
Change Staff Information
****
1. Change Home Address
2. Change Telephone Number
3. Change Username
4. Change Password
5. Change Level of Access
6. Return to Staff Menu
Enter choice:
Customer Menu
****
1. Add Customer
2. Change customer information
3. Delete customer
4. View customer
5. Return to Main Menu
Enter choice: 2
Change Customer Information
****
1. Change home address
2. Change telephone number
3. Return to Customer Menu
Enter choice:
View Customer Information
****
1. Search customer by name
2. Search customer by customer reference
3. Return to Customer Menu
Enter choice:
Quotes Menu
****
1. Add Quote
2. Change quote information
3. Delete quote
4. View quote
5. Return to Main Menu
Enter choice:
Change Quote Information
****
1. Change quote price
2. Change number of days required
3. Return to Quote Menu
Enter choice:
View Quotes
****
1. Search quote by quote reference
2. Search quote by customer name
3. Return to Quote Menu
Enter choice:
Invoice Menu
****
1. Add Invoice
2. Change invoice information
3. Delete invoice
4. View invoice
5. Return to Main Menu
Enter choice:
Change Invoice Information
****
1. Change dates worked
2. Change payment status
3. Return to Invoice Menu
Enter choice:
View Invoices
****
1. Search invoice by invoice refernece
2. Search invoice by customer reference
3. Search for unpaid invoices
4. Return to Invoice Menu
Enter choice:
Stock Menu
****
1. Add Stock
2. Change stock information
3. Delete stock
4. View stock
5. View low stock
6. Return to Main Menu
Enter choice:
Change Stock Information
****
1. Change price of stock
2. Change stock quantities
3. Return to Stock Menu
Enter choice:
View Stock Information
****
1. Search stock by stock ID
2. Sort stock quantities low to high
3. Return to Stock Menu
Enter choice:
Schedule Menu
****
1. Add Booking
2. Change booking information
3. Delete booking
4. View schedule
5. Clear all schedule
6. Return to Main Menu
Enter choice:
View Schedules
****
1. View Staff 1
2. View Staff 2
3. View Todays Work
4. Return to Schedule Menu
Enter choice:

Features that make it suitable for Stuart.

- An important thing to consider for Stuart when I was designing the UI is that it made logical sense to help guide him through the system since Stuart is used to a paper-based system the jump to computerised means it needs to be clear. To aid this all routines are in the same order across the different aspects of the system: Add, change, delete, then view. I did this to keep the system familiar so that he wasn't learning lots of different menu structures at once.
- When thinking of how to access the UI, I thought of what would help Stuart quickly access different parts of the system. For example, when he on the phone to a customer and needing to enter information, he needs to be able to quickly access the menus. Because of this, I chose a numbered systems so all he has to do is enter a single number and press enter - this means that he can quickly access routines such as adding a customer.
- The choice to use menus for the UI was because it means that I can keep it simple but clear this aids Stuart in multiple ways for example once he has familarised himself with the UI he was be able to quickly get to what he needs as he will know how to access different menus but also when he first uses the system the menu allows for clear headings so that he can easily understand what does what and where he needs to be to help the start run smoothly rather than slowing him down as he learns.

Features that make it suitable for Stuart.

- Another feature is the colour of the UI. I think that is in an important feature for Stuart as it makes it personal to him. The main colours of his company are red and white so I decided to make the text red to help keep the system personal to him and his company. Furthermore, to help the system feel personal I tried to give all the titles meaningful names so that the UI was within context for the company.
- Another feature is that if Stuart incorrectly enters something while on the UI an error message will appear to allow Stuart to understand why it hasn't worked and they can try again confidently knowing it was just a mistake and the system is not broken as Stuart is not as confident with technology and would panic if something didn’t work.
- Furthermore, since the UI is completely text based it means that it can be quickly loaded so if Stuart is starting his day and needs to find where he's working he is able to quickly load up the schedule and get to the customer as soon as.

Name of Variable	Description	Type	Scope	Routines in which it appears
loginstatus	Controls while loop for outputting the welcome, login menu	Int	Local	Main
insg	Whilever this variable is equal to zero the validation routine will run - it ensure the national insruance number is in the correct format	Int	Local	WhileMenu
ref	Whilever this variable is equal to zero the validation routine will run - it ensure the staff reference is not already in use	Int	Local	WhileMenu
fname	Whilever this variable is equal to zero the validation routine will run - it ensures a first name is entered	Int	Local	WhileMenu
lname	Whilever this variable is equal to zero the validation routine will run -it ensures a last name is entered	Int	Local	WhileMenu
adone	Whilever this variable is equal to zero the validation routine will run - it ensures that address line one is entered	Int	Local	WhileMenu
adtwo	Whilever this variable is equal to zero the validation routine will run - it ensures that address line two is entered	Int	Local	WhileMenu
adthree	Whilever this variable is equal to zero the validation routine will run - it ensures that address line three is entered	Int	Local	WhileMenu
pcode	Whilever this variable is equal to zero the validation routine will run - it ensure the postcode is in the correct format	Int	Local	WhileMenu
tel	Whilever this variable is equal to zero the validation routine will run - it ensure the mobile number entered is in the correct format	Int	Local	WhileMenu
emertel	Whilever this variable is equal to zero the validation routine will run - it ensure the mobile number entered is in the correct format	Int	Local	WhileMenu
uquser	Whilever this variable is equal to zero the validation routine will run - it ensure the username is not already in use	Int	Local	WhileMenu
passvalone	Whilever this variable is equal to zero the validation routine will run - it ensure the password conforms to a set of rules	Int	Local	WhileMenu
levrange	Whilever this variable is equal to zero the validation routine will run - it ensures the level of access is between 1 and 3	Int	Local	WhileMenu
userval	Whilever this variable is equal to zero the validation routine will run - it ensures the username is stored in the file to be able to login	Int	Local	WhileMenu
passvaltwo	Whilever this variable is equal to zero the validation routine will run - it ensures the password is stored in the file to be able to login	Int	Local	WhileMenu
mainstatus	Controls while loop for outputting the welcome, login menu	Int	Local	Menu
user	Username entered to login is passed as a parameter to function to check it is stored in the file	Char	Local	UserLoginCheck
unique	Returns a value back to the original function to say whether or not the username is found	Int	Local	UserLoginCheck
find	Used to loop through all the usernames in the file to find the one entered	Int	Local	UserLoginCheck
compare	Returns a value of whether there is a match between the username entered and one of those stored in the file	Int	Local	UserLoginCheck
pass	Password entered to login is passed as a parameter to function to check it is stored in the file	Char	Local	PassLoginCheck
unique	Returns a value back to the original function to say whether or not the password is found	Int	Local	PassLoginCheck
find	Used to loop through all the passwords in the file to find the one entered	Int	Local	PassLoginCheck

compare	Returns a value of whether there is a match between the password entered and one of those stored in the file	Int	Local	PassLoginCheck
user	Username entered to login is passed as a parameter to function to be used to find that users level of acess	Char	Local	FindLoa
find	Used to loop through all the usernames in the file to find the one entered	Char	Local	FindLoa
compare	Returns a value of whether there is a match between the username entered and one of those stored in the file	Char	Local	FindLoa
choice	Users input for a menu option on the main menu	Int	Local	MainMenu
staffstatus	Controls while loop for outputting staff menu	Int	Local	Staff
staffchoice	Users input for a menu option on the staff menu	Int	Local	StaffMenu
staffref	Unique ID for each member of staff	char	global	WhileMenu, AddStaff,UniqueStaff, SHomeAdd, STelNum, ChgLoa, ChgUsername, ChgPassword, DeleteStaff, ViewStaff , ReadBackStaffFile, ReWriteStaffFile,StaffRefCheck
fnamestaff	Staff members first name.	char	global	WhileMenu, AddStaff, SHomeAdd, STelNum, ChgLoa, ChgUsername, ChgPassword, DeleteStaff, ViewStaff, ReadBackStaffFile, ReWriteStaffFile
lnamestaff	Staff members last name.	char	global	WhileMenu, AddStaff, SHomeAdd, STelNum, ChgLoa, ChgUsername, ChgPassword, DeleteStaff, ViewStaff, ReadBackStaffFile, ReWriteStaffFile
oneadstaff	1st line of a staff members home address.	char	global	WhileMenu, AddStaff, SHomeAdd, STelNum, DeleteStaff, ViewStaff, ReadBackStaffFile, ReWriteStaffFile
twoadstaff	2nd line of a staff members home address.	char	global	WhileMenu, AddStaff, SHomeAdd, STelNum, DeleteStaff, ViewStaff, ReadBackStaffFile, ReWriteStaffFile
threeadstaff	3rd line of a staff members home address.	char	global	WhileMenu, AddStaff, SHomeAdd, STelNum, DeleteStaff, ViewStaff, ReadBackStaffFile, ReWriteStaffFile
pcodestaff	The postcode for the staff members home address	char	global	WhileMenu, AddStaff, SHomeAdd, STelNum, DeleteStaff, ViewStaff, ReadBackStaffFile, ReWriteStaffFile
telnostaff	The contact number for the member of staff.	char	global	WhileMenu, AddStaff, SHomeAdd, STelNum, ChgPassword, DeleteStaff, ViewStaff, ReadBackStaffFile, ReWriteStaffFile
emtel	The contact number for a close relative of the staff member for emergency situations	char	global	WhileMenu, AddStaff, DeleteStaff, ViewStaff, ReadBackStaffFile, ReWriteStaffFile
ninum	The national insurance number for the member of staff.	char	global	WhileMenu, AddStaff, DeleteStaff, ViewStaff, ReadBackStaffFile, ReWriteStaffFile
username	Unique username to allow a staff member to log on.	char	global	WhileMenu,AddStaff, UserLoginCheck, FindLoa,UniqueUser,ChgUsername,DeleteStaff, ReadBackStaffFile, ReWriteStaffFile
password	Unique password to allow a staff member to log on.	char	global	WhileMenu,PassLoginCheck,AddStaff,ChgPassword, DeleteStaff, ReadBackStaffFile, ReWriteStaffFile
loa	Hierarchical access to allow staff member specific access to areas of the system.	char	global	WhileMenu,AddStaff, FindLoa,ChgLoa,DeleteStaff, ReadBackStaffFile, ReWriteStaffFile

nsi	Keeps track of how many staff members are stored in the file at one time	int	global	WhileMenu, UserLoginCheck,PassLoginCheck,FindLoa,AddStaff, UniqueStaff, UniqueUser, SHomeAdd, STelNum, ChgLoa, ChgUsername, ChgPassword, DeleteStaff, ViewStaff, ReadBackStaffFile, ReWriteStaffFile,StaffRefCheck
nsc	Keeps track of how many staff members are stored in the file at one time	char	global	WhileMenu,AddStaff,DeleteStaff, ReadBackStaffFile, ReWriteStaffFile
isfirsttime	When logging in finds if the staff file is empty and if so brings up the information to add a staff member	int	global	Login, WhileMenu
level	A users specific level of access which is used throughout the program to see if they have access to certain parts of the system	int	global	FindLoa,MainMenu, ChangeStaffMenu, CustomerMenu,QuotesMenu,InvoiceMenu,StockMenu,ScheduleMenu,
insg	Whilever this variable is equal to zero the validation routine will run - it ensure the national insruance number is in the correct format	Int	Local	AddStaff
ref	Whilever this variable is equal to zero the validation routine will run - it ensure the staff reference is not already in use	Int	Local	AddStaff
fname	Whilever this variable is equal to zero the validation routine will run - it ensures a first name is entered	Int	Local	AddStaff
lname	Whilever this variable is equal to zero the validation routine will run -it ensures a last name is entered	Int	Local	AddStaff
adone	Whilever this variable is equal to zero the validation routine will run - it ensures that address line one is entered	Int	Local	AddStaff
adtwo	Whilever this variable is equal to zero the validation routine will run - it ensures that address line two is entered	Int	Local	AddStaff
adthree	Whilever this variable is equal to zero the validation routine will run - it ensures that address line three is entered	Int	Local	AddStaff
pcode	Whilever this variable is equal to zero the validation routine will run - it ensure the postcode is in the correct format	Int	Local	AddStaff
tel	Whilever this variable is equal to zero the validation routine will run - it ensure the mobile number entered is in the correct format	Int	Local	AddStaff
emertel	Whilever this variable is equal to zero the validation routine will run - it ensure the mobile number entered is in the correct format	Int	Local	AddStaff
uquser	Whilever this variable is equal to zero the validation routine will run - it ensure the username is not already in use	Int	Local	AddStaff
passvalone	Whilever this variable is equal to zero the validation routine will run - it ensure the password conforms to a set of rules	Int	Local	AddStaff
levrange	Whilever this variable is equal to zero the validation routine will run - it ensures the level of access is between 1 and 3	Int	Local	AddStaff
bufferstaff	Returns a value back to the original function to say whether or not the staff reference is the correct amount of characters.	Int	Local	AddStaff
telbuff	Returns a value back to the original function to say whether or not the mobile number is the correct amount of characters.	Int	Local	AddStaff
emtelbuff	Returns a value back to the original function to say whether or not the mobile number is the correct amount of characters.	Int	Local	AddStaff
nibuff	Returns a value back to the original function to say whether or not the national insurance number is the correct amount of characters.	Int	Local	AddStaff
emtelb	Mobile number entered is passed as a parameter to see if its the correct number of characters	Char	Local	AddStaff

staffb	Staff referebce entered is passed as a parameter to see if its the correct number of characters	Char	Local	AddStaff
bni	National insurance number entered is passed as a parameter to see if its the correct number of characters	Char	Local	AddStaff
telb	Mobile number entered is passed as a parameter to see if its the correct number of characters	Char	Local	AddStaff
staff	Staff reference is passed as a parameter to check that the reference entered is unique	Char	Local	UniqueStaff
uniqueref	Returns a value back to the original function to say whether or not the staff reference is unique	Char	Local	UniqueStaff
find	Used to loop through all the references in the staff file to see if it can find the one entered	Char	Local	UniqueStaff
compare	Returns a value of whether there is a match beween the staff reference entered and one stored in the file	Char	Local	UniqueStaff
len	Stores the number of characters entered from the input to check data has been entered	Int	Local	UniqueStaff
position	Used to loop through all the characters in the reference to check no characters are entered	Int	Local	UniqueStaff
presence	First name is passed as a parameter to check that data has been entered	Char	Local	PresVal
length	Stores the number of characters entered from the input to check data has been entered	Int	Local	PresVal
presence	Last name is passed as a parameter to check that data has been entered	Char	Local	PresValL
length	Stores the number of characters entered from the input to check data has been entered	Int	Local	PresValL
presence	Address line 1 is passed as a parameter to check that data has been entered	Char	Local	PresValO
length	Stores the number of characters entered from the input to check data has been entered	Int	Local	PresValO
presence	Address line 2 is passed as a parameter to check that data has been entered	Char	Local	PresValT
length	Stores the number of characters entered from the input to check data has been entered	Int	Local	PresValT
presence	Address line 3 is passed as a parameter to check that data has been entered	Char	Local	PresValTh
length	Stores the number of characters entered from the input to check data has been entered	Int	Local	PresValTh
postcode	Postcode entered is passed as a parameter for validation checking	Char	Local	PostCodeVal
valid	Returns a value back to the original function to say whether or not the postcode is the correct format	Int	Local	PostCodeVal
length	Stores the number of characters entered from the input to check data has been entered	Int	Local	PostCodeVal
phonenum	Mobile number entered is passed as a parameter for validation checking	Char	Local	TelVal
valid	Returns a value back to the original function to say whether or not the mobile number is the correct format	Int	Local	TelVal
length	Stores the number of characters entered from the input to check data has been entered and to check that the length of the input is correct	Int	Local	TelVal
position	Used to loop through all the characters in the phone number enteredto check they either start 07 or are just numbers	Int	Local	TelVal
phonenum	Mobile number entered is passed as a parameter for validation checking	Char	Local	TelVal
valid	Returns a value back to the original function to say whether or not the mobile number is the correct format	Int	Local	TelVal
length	Stores the number of characters entered from the input to check data has been entered and to check that the length of the inout is correct	Int	Local	TelVal
position	Used to loop through all the characters in the phone number enteredto check they either start 07 or are just numbers	Int	Local	TelVal

ninum	National insurance number entered is passed as a parameter for validation checking	Char	Local	ValidNI
valid	Returns a value back to the original function to say whether or not the national insurance number is the correct format	Int	Local	ValidNI
len	Stores the number of characters entered from the input to check data has been entered and to check that the length of the input is correct	Int	Local	ValidNI
user	Username entered is passed as a parameter to the function to check the username is unique and not already stored in the file	Char	Local	UniqueUser
userok	Returns a value back to the original function to say whether or not the username is unqiue	Int	Local	UniqueUser
find	Used to loop through all the usernames in the staff file to see if there is a match	Int	Local	UniqueUser
compare	Returns a value of whether there is a match between the username entered and one stored in the file	Int	Local	UniqueUser
len	Stores the number of characters entered from the input to check data has been entered	Int	Local	UniqueUser
pass	Password entered is passed as a parameter for validation	Char	Local	PasswordVal
valid	Returns a value back to the original function to say whether or not the password is the correct format	Int	Local	PasswordVal
length	Stores the number of characters entered from the input to check data has been entered and to check that the length of the input is correct	Int	Local	PasswordVal
checkpunc	Used to loop through all the characters in the password to check that at least one is a piece of punctuation	Int	Local	PasswordVal
level	Level of access entered is passed as a paramter to check its within a certain range	Char	Local	RangeCheck
valid	Returns a value back to the original function to say whether or not the access level is in the correct range	Int	Local	RangeCheck
intlev	Used to convert the entered level of access to an integer for the range check	Int	Local	RangeCheck
len	Stores the number of characters entered from the input to check data has been entered	Int	Local	RangeCheck
position	Used to loop through all the characters in the access level to check no characters are entered	Int	Local	RangeCheck
changestatus	Controls while loop for outputtin change staff menu	Int	Local	ChangeStaff
changestaffchoice	User input for menu option on change staff menu	Int	Local	ChangeStaffMenu
looking	Staff reference inputted by the user	Char	Local	SHomeAdd
compare	Returns a value of whether there is a match between the staff reference entered and one stored in the file	Int	Local	SHomeAdd
find	Used to loop through all the references in the staff file to see if there is a match	Int	Local	SHomeAdd
result	Stores the value of the confirmation of whether or not it is the correct staff members details to change	Int	Local	SHomeAdd
compresult	Returns a value of whether there is a match between the input of Y or N entered and Y to check whether they can continue	Int	Local	SHomeAdd
adone	Whilever this variable is equal to zero the validation routine will run - it ensures that address line one is entered	Int	Local	SHomeAdd
adthree	Whilever this variable is equal to zero the validation routine will run - it ensures that address line three is entered	Int	Local	SHomeAdd
pcode	Whilever this variable is equal to zero the validation routine will run - it ensure the postcode is in the correct format	Int	Local	SHomeAdd
looking	Staff reference inputted by the user	Char	Local	STelNum

compare	Returns a value of whether there is a match between the staff reference entered and one stored in the file	Int	Local	STelNum
find	Used to loop through all the references in the staff file to see if there is a match	Int	Local	STelNum
result	Stores the value of the confirmation of whether or not it is the correct staff members details to change	Int	Local	STelNum
compresult	Returns a value of whether there is a match between the input of Y or N entered and Y to check whether they can continue	Int	Local	STelNum
tel	Whilever this variable is equal to zero the validation routine will run - it ensures that the mobile number is in the correct format	Int	Local	STelNum
looking	Staff reference inputted by the user	Char	Local	ChgLoa
compare	Returns a value of whether there is a match between the staff reference entered and one stored in the file	Int	Local	ChgLoa
find	Used to loop through all the references in the staff file to see if there is a match	Int	Local	ChgLoa
result	Stores the value of the confirmation of whether or not it is the correct staff members details to change	Int	Local	ChgLoa
compresult	Returns a value of whether there is a match between the input of Y or N entered and Y to check whether they can continue	Int	Local	ChgLoa
levrange	Whilever this variable is equal to zero the validation routine will run - it ensures that the access level is in the correct range	Int	Local	ChgLoa
looking	Staff reference inputted by the user	Char	Local	ChgUsername
compare	Returns a value of whether there is a match between the staff reference entered and one stored in the file	Int	Local	ChgUsername
compresult	Returns a value of whether there is a match between the input of Y or N entered and Y to check whether they can continue	Int	Local	ChgUsername
find	Used to loop through all the references in the staff file to see if there is a match	Int	Local	ChgUsername
result	Stores the value of the confirmation of whether or not it is the correct staff members details to change	Int	Local	ChgUsername
uquser	Whilever this variable is equal to zero the validation routine will run - it ensures the username is unique	Int	Local	ChgUsername
newusername	Stores the new username entered by the user to be passed to the validation routine	Int	Local	ChgUsername
looking	Staff reference inputted by the user	Char	Local	ChgPassword
compare	Returns a value of whether there is a match between the staff reference entered and one stored in the file	Int	Local	ChgPassword
compresult	Returns a value of whether there is a match between the input of Y or N entered and Y to check whether they can continue	Int	Local	ChgPassword
find	Used to loop through all the references in the staff file to see if there is a match	Int	Local	ChgPassword
result	Stores the value of the confirmation of whether or not it is the correct staff members details to change	Int	Local	ChgPassword
passval	Whilever this variable is equal to zero the validation routine will run - it ensures the password is within a correct format	Int	Local	ChgPassword
find	Used to loop through all the references in the staff file to see if there is a match	Int	Local	DeleteStaff
compare	Returns a value of whether there is a match between the staff reference entered and one stored in the file	Int	Local	DeleteStaff
compresult	Returns a value of whether there is a match between the input of Y or N entered and Y to check whether they can continue	Int	Local	DeleteStaff

del	Used to loop through the file and move all positions by -1	int	Local	DeleteStaff
result	Stores the value of the confirmation of whether or not it is the correct staff members details to change	Int	Local	DeleteStaff
looking	Staff reference inputted by the user	Char	Local	DeleteStaff
looking	Staff reference inputted by the user	Char	Local	ViewStaff
compare	Returns a value of whether there is a match between the staff reference entered and one stored in the file	Int	Local	ViewStaff
find	Used to loop through all the references in the staff file to see if there is a match	Int	Local	ViewStaff
confirm	Use enters Y or N to say if they want all files to be recovered	Int	Local	Recovery
compare	Returns a value whether there is a match between the confirmation entered and 'Y'	Int	Local	Recovery
count	Used to loop through all variables stored in the staff file	Int	Local	ReadBackStaffFile
count	Used to loop through all variables stored in the staff file	Int	Local	ReWriteStaffFile
title	Title of customer i.e Mr, Miss	Char	Global	AddCustomer,CName,CRef
fnamecust	Customers first name	Char	Global	AddCustomer,CHomeAdd,CTelNum,DeleteCustomer,CName,CRef,LocateCust,LocateCustInvoice
lnamecust	Customers last name	Char	Global	AddCustomer,CHomeAdd,CTelNum,DeleteCustomer,CName,CRef,LocateCust,LocateCustInvoice
oneadcust	Address line 1 of customers home address	Char	Global	AddCustomer,CHomeAdd,CName,CRef
twoadcust	Address line 2 of customers home address	Char	Global	AddCustomer,CHomeAdd,CName,CRef
threeadcust	Address line 3 of customers home address	Char	Global	AddCustomer,CHomeAdd,CName,CRef
pcodecust	Postcode of customers home address	Char	Global	AddCustomer,CHomeAdd,CName,CRef
telnocust	Customer contact number	Char	Global	AddCustomer,CHomeAdd,CTelNum,CName,CRef,LocateCust,LocateCustInvoice
flag	Stores whether or not the record is blank or not	Char	Global	AddCustomer,CHomeAdd,CTelNum,DeleteCustomer,CName,CRef,LocateCustQVal,LocateCust,LocateCustIVal,LocateCustInvoice
custref	Unique reference of each customer	Int	Global	AddCustomer,CHomeAdd,CTelNum,DeleteCustomer,CName,CRef,LocateCustQVal,LocateCust,LocateCustIVal,LocateCustInvoice,LocateCustSchedule
customerstatus	Controls while loop for outputting customer menu	Int	Local	Customers
customerchoice	User input for menu option on customer menu	Int	Local	CustomerMenu
tpres	Whilever this variable is equal to zero the validation routine will run - it ensure that data has been entered	Int	Local	AddCustomer
fname	Whilever this variable is equal to zero the validation routine will run - it ensure that data has been entered	Int	Local	AddCustomer
lname	Whilever this variable is equal to zero the validation routine will run - it ensure that data has been entered	Int	Local	AddCustomer
adone	Whilever this variable is equal to zero the validation routine will run - it ensure that data has been entered	Int	Local	AddCustomer
adthree	Whilever this variable is equal to zero the validation routine will run - it ensure that data has been entered	Int	Local	AddCustomer
post	Whilever this variable is equal to zero the validation routine will run - it ensure the postcode is of the correct format	Int	Local	AddCustomer

tel	Whilever this variable is equal to zero the validation routine will run - it ensure the mobile number is of the correct format	Int	Local	AddCustomer
ref	Whilever this variable is equal to zero the validation routine will run - it ensure that the customer reference doesn't already exists	Int	Local	AddCustomer
tempcust	Stores customer reference as a character for validation checks	Char	Local	AddCustomer
telb	Mobile number entered is passed as a parameter to see if its the correct number of characters	Char	Local	AddCustomer
telbuff	Returns a value back to the original function to say whether or not the mobile number is the correct amount of characters.	Int	Local	AddCustomer
cust	Customer reference is passed as a parameter to see if its already in use or not	Char	Local	CustRefVal
length	Stores the number of characters entered from the input to check data has been entered	Int	Local	CustRefVal
valid	Returns a value back to the original function to say whether or not the customer reference can be used or not	Int	Local	CustRefVal
position	Used to loop through all the characters in the reference entere dto check it's a number	Int	Local	CustRefVal
compare	Returns a value of whether the record space to enter the new customer reference is free or not	Int	Local	CustRefVal
presence	Title is passed as a parameter to check that data has been entered	char	Local	PresValTitle
length	Stores the number of characters entered from the input to check data has been entered	Int	Local	PresValTitle
presence	First name entered is passed as a parameter to check that data has been entered	char	Local	PresValFirst
length	Stores the number of characters entered from the input to check data has been entered	Int	Local	PresValFirst
presence	Last name entered is passed as a parameter to check that data has been entered	char	Local	PresValLast
length	Stores the number of characters entered from the input to check data has been entered	Int	Local	PresValLast
presence	Address line one entered is passed as a parameter to check that data has been entered	char	Local	PresValOne
length	Stores the number of characters entered from the input to check data has been entered	Int	Local	PresValOne
presence	Address line 2 entered is passed as a parameter to check that data has been entered	char	Local	PresValTwo
length	Stores the number of characters entered from the input to check data has been entered	Int	Local	PresValTwo
presence	Address line 3 entered is passed as a parameter to check that data has been entered	char	Local	PresValThree
length	Stores the number of characters entered from the input to check data has been entered	Int	Local	PresValThree
valid	Returns a value back to the original function to say whether or not the postcode is in the correct format	Int	Local	PostcodeValCust
length	Stores the number of characters entered from the input to check data has been entered and is of the correct length	Int	Local	PostcodeValCust
postcode	Postcode entered is passed as a parameter to check that data has been entered and to check it is of the correct format	Char	Local	PostcodeValCust
phonenum	Phone number entered is passed as a parameter to check that data has been entered and to check it is of the correct format	Char	Local	TelValCust
valid	Returns a value back to the original function to say whether or not the phone number is in the correct format	Int	Local	TelValCust
len	Stores the number of characters entered from the input to check data has been entered and is of the correct length	Int	Local	TelValCust
position	Used to loop through all the characters in the phone number enteredto check they either start 07 or are just numbers	Int	Local	TelValCust

changestatus	Controls while loop for outputting customer change menu	Int	Local	ChangeCustomer
changecustomerchoice	User input for menu option on change customer menu	Int	Local	ChangeCustomerMenu
name	Customers last name entered by the user to find them to change their address	Char	Local	CHomeAdd
compare	Returns a value of whether there is a match between the last name of the customer entered and one stored in the file	Int	Local	CHomeAdd
result	Stores the value of the confirmation of whether or not it is the correct customers details to change	Int	Local	CHomeAdd
adone	Whilever this variable is equal to zero the validation routine will run - it ensures that address line one is entered	Int	Local	CHomeAdd
compresult	Returns a value of whether there is a match between the input of Y or N entered and Y to check whether they can continue	Int	Local	CHomeAdd
adthree	Whilever this variable is equal to zero the validation routine will run - it ensures that address line three is entered	Int	Local	CHomeAdd
post	Whilever this variable is equal to zero the validation routine will run - it ensure the postcode is in the correct format	Int	Local	CHomeAdd
len	Stores the number of characters entered from the input to check data has been entered and is of the correct length	Int	Local	CHomeAdd
name	Customers last name entered by the user to find them to change their telephone number	Char	Local	CTelNum
compare	Returns a value of whether there is a match between the last name of the customer entered and one stored in the file	Int	Local	CTelNum
result	Stores the value of the confirmation of whether or not it is the correct customers details to change	Int	Local	CTelNum
tel	Whilever this variable is equal to zero the validation routine will run - it ensures that the telephone number is entered as is of the correct format	Int	Local	CTelNum
compresult	Returns a value of whether there is a match between the input of Y or N entered and Y to check whether they can continue	Int	Local	CTelNum
len	Stores the number of characters entered from the input to check data has been entered and is of the correct length	Int	Local	CTelNum
compare	Returns a value of whether the record space is used up for a customer to be deleted or if its already empty	Int	Local	DeleteCustomer
result	Stores the value of the confirmation of whether or not it is the correct customers details to delete	Int	Local	DeleteCustomer
compresult	Returns a value of whether there is a match between the input of Y or N entered and Y to check whether they can continue	Int	Local	DeleteCustomer
len	Stores the number of characters entered from the input to check data has been entered and is of the correct length	Int	Local	DeleteCustomer
charref	Stores customer reference as a character for validation checks	Char	Local	DeleteCustomer
cust	Customer reference is passed as a parameter to check to see if they have a booking on the schedule	Int	Local	ScheduleCheck
find	Used to loop through all the customer references in the quote file to see if there is a match	Int	Local	ScheduleCheck
compare	Returns a value of whether there is a match between the customer reference entered and one stored in the file	Int	Local	ScheduleCheck
custchar	Used to store customer reference as a character	Char	Local	ScheduleCheck
viewstatus	Controls while loop for outputting customer view menu	Int	Local	ViewCustomer

viewcustomerchoice	User input for menu option on view customer menu	Int	Local	ViewCustomerMenu
compare	Returns a value of whether there is a match between the last name of the customer entered and one stored in the file	Int	Local	CName
cmp	Returns a value of whether the flag shows the record space to be filled or empty	Int	Local	CName
name	Customer last name entered by the user	Char	Local	CName
len	Stores the number of characters entered from the input to check data has been entered and is of the correct length	Int	Local	CName
compare	Returns a value of whether the flag shows the record space to be filled or empty	Int	Local	CRef
charref	Used to store customer reference as a character	Char	Local	CRef
len	Stores the number of characters entered from the input to check data has been entered and is of the correct length	Int	Local	CRef
quoteref	Unique reference of each quote	Char	Global	AddQuote,QPrice,QDay,DeleteQuote,QRef,QuoteRefCheck,LocateQuoteCheck,LocateQuote,ReadBackQuoteFile,ReWriteQuoteFile,QuoteRefCheckBooking,LocateQuoteWork
custno	Stores customer reference to link customer information to the quote	Char	Global	AddQuote,QPrice,QDay,DeleteQuote,QRef,QCustRef,ReadBackQuoteFile,ReWriteQuoteFile,LocateQuoteWork
quotedate	Date the quote was produced	Char	Global	AddQuote,DeleteQuote,QCustRef,ReadBackQuoteFile, ReWriteQuoteFile
mainjobdesc	The job description for what the customer wants doing when they ring up	Char	Global	AddQuote,QPrice,QDay,DeleteQuote,QRef,QCustRef,LocateQuoteCheck,LocateQuote,ReadBackQuoteFile,ReWriteQuoteFile,LocateQuoteWork
numofdays	Number of days required for the quote	Char	Global	AddQuote,QDay,DeleteQuote,QRef,QCustRef,ReadBackQuoteFile,ReWriteQuoteFile
labourq	Labour cost for the quote	Char	Global	AddQuote,QDay,DeleteQuote,QRef,QCustRef,LocateQuote,ReadBackQuoteFile,ReWriteQuoteFile
mileage	Price of travelling to the home	Char	Global	AddQuote,DeleteQuote,QRef,QCustRef,LocateQuote,ReadBackQuoteFile,ReWriteQuoteFile
vat	20% VAT is calculated to be added to the total cost	Char	Global	AddQuote,DeleteQuote,LocateQuote,ReadBackQuoteFile, ReWriteQuoteFile
stockcost	Price of materials for the quote	Char	Global	AddQuote,DeleteQuote,QRef,QCustRef,LocateQuote,ReadBackQuoteFile,ReWriteQuoteFile
totalcost	Overall price of the quote	Char	Global	AddQuote,QPrice,QDay,DeleteQuote,QRef,QCustRef,LocateQuote,ReadBackQuoteFile, ReWriteQuoteFile
nqi	Keeps track of how many quotes are stored in the file at one time	Int	Global	AddQuote, UniqueQ,QPrice,QDay,DeleteQuote,QRef,QCustRef,QuoteRefCheck,LocateQuoteCheck,LocateQuote,ReadBackQuoteFile, ReWriteQuoteFile,QuoteRefCheckBooking,LocateQuoteWork
nqc	Keeps track of how many quotes are stored in the file at one time	Char	Global	AddQuote,DeleteQuote,ReadBackQuoteFile, ReWriteQuoteFile
yeart	Used to extract and store the year from the systems clock	Int	Global	SystemsClockBooking
monthchar	Used to extract and store the month from the systems clock	Char	Global	SystemsClockBooking
dayst	Used to extract and store the day from the systems clock	Int	Global	SystemsClockBooking
rawtime	Used to find the time stored on the systems clock	Int	Global	SystemsClockBooking
temptime	Stores variable 'rawtime' as a string for manipulation	Char	Global	SystemsClockBooking
monthval	Returns a value of whether there is a match between the month extracted and the one stored	Int	Global	SystemsClockBooking

monthint	Stores the value of the month as an integer	Int	Global	SystemsClockBooking
quotestatus	Controls while loop for outputting quote menu	Int	Local	Quotes
quotechoice	User input for menu option on quote menu	Int	Local	QuoteMenu
numofitems	Stores how many items of stock are required for the quote	Int	Local	AddQuote
count	Used to loop through the number of items of stock required and find there prices	Int	Local	AddQuote
quotenum	Stores the quote reference entered by the user as an integer value for saving it to the links file	Int	Local	AddQuote
stocktotal	Running total of all stock once added together	Int	Local	AddQuote
cost	Individual price of an item of stock which is then added to the stock total	Int	Local	AddQuote
numofdaysq	Saves number of days entered by the user as a integer to be used in the calculation - for labour prices	Int	Local	AddQuote
mileageq	Saves mileage cost entered by the user as an integer to be used in the calculation	Int	Local	AddQuote
VAT	Result of a calculation that determines the vat to be added to the total cost	Int	Local	AddQuote
labour	Result of a calculation that determines the labour prices depending on the number of days	Int	Local	AddQuote
totoal	Result of a calculation that determines the total price of the quote, adding together all the different aspects	Int	Local	AddQuote
qref	Whilever this variable is equal to zero the validation routine will run - it ensures that the quote reference entered is unique	Int	Local	AddQuote
cust	Whilever this variable is equal to zero the validation routine will run - it ensures that the customer reference entered exists	Int	Local	AddQuote
dateq	Whilever this variable is equal to zero the validation routine will run - it ensures that the date entered is in the correct format	Int	Local	AddQuote
presm	Whilever this variable is equal to zero the validation routine will run - it ensures that data has been entered	Int	Local	AddQuote
dayrange	Whilever this variable is equal to zero the validation routine will run - it ensures that data entered is between a certain range	Int	Local	AddQuote
prestravel	Whilever this variable is equal to zero the validation routine will run - it ensures that data has been entered	Int	Local	AddQuote
lstock	Stock reference entered by the user is passed as a paramter to find the price of the item of stock to add to the running total for materials	Char	Local	FindStock
find	Used to loop through all the references in the stock file to see if there is a match	Int	Local	FindStock
compare	Returns a value of whether there is a match between the stock reference entered and one stored in the file	Int	Local	FindStock
cost	Stores a running total of the prices of all the materials as they are calculated to return to the calculation in add quote	Int	Local	FindStock
quant	Stores the input from the user of how many of an item of stock they want	Int	Local	FindStock
sp	Used to save the price of an item of stock as an integer for calculations	Int	Local	FindStock
quote	The quote reference the user entered in add quote is passed as a parameter to see if its unique	Char	Local	UniqueQ
uniqueref	Returns a value back to the original function to say whether or not the quote reference is unqiue	Int	Local	UniqueQ
find	Used to loop through all the references in the quote file to see if there is a match	Int	Local	UniqueQ
compare	Returns a value of whether there is a match between the quote reference entered and one stored in the file	Int	Local	UniqueQ
len	Stores the number of characters entered from the input to check data has been entered	Int	Local	UniqueQ

custno	Customer reference is passed as a parameter to check if the customer exists	Char	Local	LocateCustQVal
compare	Returns a value of whether the record space is in use or not	Int	Local	LocateCustQVal
valid	Returns a value back to the original function to say whether or not the customer exists	Int	Local	LocateCustQVal
date	Date entered by the user is passed as a parameter to check its in the correct format	Char	Local	DateVal
valid	Returns a value back to the original function to say whether or not the date is in the correct format	Int	Local	DateVal
months	Used to extract the month as an integer from the date entered to use to check if the days are correct	Int	Local	DateVal
days	Used to extract the day as an integer from the date entered to do a range check	Int	Local	DateVal
years	Used to extract the year as a float from the date to used is a calculation to check if it is a leap year or not	Float	Local	DateVal
len	Stores the number of characters entered from the input to check data is in the correct format and that data has been entered	Int	Local	DateVal
presence	Job description is passed as a parameter to check that data has been entered	Char	Local	PresValJ
length	Stores the number of characters entered from the input to check data has been entered	Int	Local	PresValJ
day	Number of days entered on the job is passed as a parameter for a range check	Char	Local	RangeDay
valid	Returns a value back to the original function to say whether or not the number of days is within the range	Int	Local	RangeDay
intday	Used to convert the number of days to an integer to be used in the range check	Int	Local	RangeDay
presence	Mileage cost is passed as a parameter to check that data has been entered	Char	Local	PresValTr
length	Stores the number of characters entered from the input to check data has been entered	Int	Local	PresValTr
quotestatus	Controls while loop for outputting change quote menu	Int	Local	ChangeQuote
changequotechoice	User input for menu option on change quote menu	Int	Local	ChangeQuote
looking	Quote reference entered by the user	Char	Local	QPrice
compare	Returns a value of whether there is a match between the quote reference entered and one stored in the file	Int	Local	QPrice
find	Used to loop through all the references in the quote file to see if there is a match	Int	Local	QPrice
result	Stores the value of the confirmation of whether or not it is the correct quote to change the price of	Int	Local	QPrice
priceval	Whilever this variable is equal to zero the validation routine will run - it ensures that the price entered is within a reasonable range	Int	Local	QPrice
cost	New price entered from the change is passed as a parameter to check that its within a suitable range	Char	Local	QuotePriceRange
price	Stores 'cost' as an integer to do the range check	Int	Local	QuotePriceRange
valid	Returns a value back to the original function to say whether or not the price is within the range	Int	Local	QuotePriceRange
len	Stores the number of characters entered from the input to check data has been entered	Int	Local	QuotePriceRange
looking	Quote reference entered by the user	Char	Local	QDay
compare	Returns a value of whether there is a match between the quote reference entered and one stored in the file	Int	Local	QDay
find	Used to loop through all the references in the quote file to see if there is a match	Int	Local	QDay
result	Stores the value of the confirmation of whether or not it is the correct quote to change the days worked	Int	Local	QDay

numofdaysq	Used to convert the number of days to an integer to be used in a calculation to work out new labour prices	Int	Local	QDay
newlabour	Result of calculation to work out updated labour cost	Int	Local	QDay
oldlabour	Stores current labour price of quote as an integer to calculate the difference in the two prices	Int	Local	QDay
difference	Works out the extra cost of labour prices by working out the difference to add it to the total cost	Int	Local	QDay
totali	Used to store the current total price as an integer for calculations	Int	Local	QDay
newtotal	Result of calculation for total cost of the quote with updated labour cost	Int	Local	QDay
dayrange	Whilever this variable is equal to zero the validation routine will run - it ensures that the days entered is within a reasonable range	Int	Local	QDay
find	Used to loop through all the references in the quote file to see if there is a match	Int	Local	DeleteQuote
compare	Returns a value of whether there is a match between the quote reference entered and one stored in the file	Int	Local	DeleteQuote
del	Used to loop through the file and move all positions by -1	Int	Local	DeleteQuote
looking	Quote reference entered by the user	Char	Local	DeleteQuote
result	Stores the value of the confirmation of whether or not it is the correct quote to delete	Int	Local	DeleteQuote
viewstatus	Controls while loop for outputting view quote menu	Int	Local	ViewQuote
viewquotechoice	User input for menu option on view quote menu	Int	Local	ViewQuoteMenu
looking	Quote reference entered by the user	Char	Local	QRef
compare	Returns a value of whether there is a match between the quote reference entered and one stored in the file	Int	Local	QRef
find	Used to loop through all the references in the quote file to see if there is a match	Int	Local	QRef
pound	Used to output a £ sign on the price breakdown	Char	Local	QRef
custno	Customer reference found in the quote is passed as a parameter to output further customer information when viewing a quote	Char	Local	LocateCust
compare	Returns a value of whether the record space is in use or not	Int	Local	LocateCust
compare	Returns a value of whether there is a match between the customer reference entered and one stored in the file	Int	Local	QCustRef
find	Used to loop through all the references in the quote file to see if there is a match	Int	Local	QCustRef
looking	Customer reference entered by the user	Char	Local	QCustRef
pound	Used to output a £ sign on the price breakdown	Char	Local	QCustRef
count	Used to loop through all variables stored in the quote file	Int	Local	ReadBackSQuotesFile
count	Used to loop through all variables stored in the quote file	Int	Local	ReWriteQuotesFile
invoiceref	Unique reference for each invoice	Char	Global	AddInvoice,UniqueInvoice,IDates,IPay,DeleteInvoice,IRef,ReadBackInvoiceFile, ReWriteInvoiceFile
custnum	Customer reference linked to the invoice	Char	Global	AddInvoice,IDates,IPay,DeleteInvoice,IRef,ICustRef,IUnpaid,ReadBackInvoiceFile, ReWriteInvoiceFile
quotenum	Quote reference linked to the invoice	Char	Global	AddInvoice,IDates,IPay,DeleteInvoice,IRef,ICustRef,ReadBackInvoiceFile, ReWriteInvoiceFile
invoicedate	Date that the invoice was produced	Char	Global	AddInvoice,DeleteInvoice,IRef,ICustRef, ReadBackInvoiceFile, ReWriteInvoiceFile
jobstartdate	Date that the booking started	Char	Global	AddInvoice,IDates,DeleteInvoice,IRef,ICustRef,ReadBackInvoiceFile, ReWriteInvoiceFile

jobenddate	Date that the booking ended	Char	Global	AddInvoice,IDates,DeleteInvoice,IRef,ICustRef,ReadBackInvoiceFile, ReWriteInvoiceFile
paid	States whether or not the invoice is paid for	Char	Global	AddInvoice,IPay,DeleteInvoice,IRef,ICustRef,IUnpaid,ReadBackInvoiceFile, ReWriteInvoiceFile
nii	Stores the value for the number of invoices stored in the file	Int	Global	AddInvoice,UniqueInvoice,IDates,IPay,DeleteInvoice,IRef,ICustRef,IUnpaid,ReadBackInvoiceFile, ReWriteInvoiceFile
nic	Stores the value for the number of invoices stored in the file	Char	Global	AddInvoice,DeleteInvoice,ReadBackInvoiceFile, ReWriteInvoiceFile
monthstart	Extracts the month from the start date to be used elsewhere	Int	Global	DateValStart, RangeDate
yearstart	Extracts the year from the start date to be used elsewhere	Int	Global	DateValStart, RangeDate
daystart	Extracts the day from the start date to be used elsewhere	Int	Global	DateValStart, RangeDate
invoicestatus	Controls while loop for outputting invoice menu	Int	Local	Invoice
invoicechoice	User input for menu option on invoice menu	Int	Local	InvoiceMenu
ref	Whilever this variable is equal to zero the validation routine will run - it ensures that the invoice reference entered is unique	Int	Local	Add Invoice
cref	Whilever this variable is equal to zero the validation routine will run - it ensures that the customer reference entered exists within the system	Int	Local	Add Invoice
qref	Whilever this variable is equal to zero the validation routine will run - it ensures that the quote reference entered exists within the system	Int	Local	Add Invoice
start	Whilever this variable is equal to zero the validation routine will run - it ensures that the start date entered is of the correct format	Int	Local	Add Invoice
endrange	Whilever this variable is equal to zero the validation routine will run - it ensures that the end date entered is after the start date	Int	Local	Add Invoice
end	Whilever this variable is equal to zero the validation routine will run - it ensures that the end date entered is of the correct format.	Int	Local	Add Invoice
payment	Whilever this variable is equal to zero the validation routine will run - it ensures that the payment status entered is either 0 or 1	Int	Local	Add Invoice
invoice	Invoice reference is passed as a parameter to check that the reference is unique	Char	Local	UniqueInvoice
uniqueref	Returns a value back to the original function to say whether or not the invoice reference is unique	Int	Local	UniqueInvoice
find	Used to loop through all the references in the invoice file to see if there is a match	Int	Local	UniqueInvoice
compare	Returns a value of whether there is a match between the invoice reference entered and one stored in the file	Int	Local	UniqueInvoice
len	Stores the number of characters entered from the input to check data has been entered	Int	Local	UniqueInvoice
custnum	Customer reference is passed as a parameter to check that the reference exists	Char	Local	LocateCustIVal
compare	Returns a value of whether the record space is in use or not	Int	Local	LocateCustIVal
valid	Returns a value back to the original function to say whether or not the customer reference exists within the system	Int	Local	LocateCustIVal
quote	Quote reference is passed as a parameter to check that the reference exists	Char	Local	QuoteRefCheck
uniqueref	Returns a value back to the original function to say whether or not the quote reference exists within the system	Int	Local	QuoteRefCheck
find	Used to loop through all the references in the quotes file to see if there is a match	Int	Local	QuoteRefCheck

compare	Returns a value of whether there is a match between the quote reference entered and one stored in the file	Int	Local	QuoteRefCheck
date	Start date is passed as a parameter to check that it is of the correct format	Char	Local	DateValStart
valid	Returns a value back to the original function to say whether or not the date is of the correct format	Int	Local	DateValStart
len	Stores the number of characters entered from the input to check data has been entered	Int	Local	DateValStart
date	End date is passed as a parameter to check that it is of the correct format	Char	Local	DateValEnd
valid	Returns a value back to the original function to say whether or not the date is of the correct format	Int	Local	DateValEnd
months	Extracts the month from the date entered to check that the day entered is in the correct range	Int	Local	DateValEnd
days	Extracts the day entered from the date for validating	Int	Local	DateValEnd
years	Extracts the year to see if it is a leap year for furtrher validation	Float	Local	DateValEnd
len	Stores the number of characters entered from the input to check data has been entered	Int	Local	DateValEnd
compare	End date is passed as a parameter to check that it is after the start date	Char	Local	RangeDate
day	Extracts the day entered from the date for validating	Int	Local	RangeDate
month	Extracts the month from the date entered	Int	Local	RangeDate
year	Extracts the year from the date entered	Int	Local	RangeDate
valid	Returns a value back to the original function to say whether or not the date is after the start date	Int	Local	RangeDate
pay	Payment status is passed as a parameter to check that the data entered is within a certain range	Char	Local	RangePaid
valid	Returns a value back to the original function to say whether or not the payment status is in the correct range	Int	Local	RangePaid
intpay	Used to convert the payment status entered to an integer for the range check	Int	Local	RangePaid
changestatus	Controls while loop for outputting invoice change menu	Int	Local	ChangeInvoice
changeinvoicechoice	User input for menu option on invoice change menu	Int	Local	ChangeInvoiceMenu
looking	Invoice reference entered by the user to have the dates changed	Char	Local	IDates
compare	Returns a value of whether there is a match between the invoice reference entered and one stored in the file	Int	Local	IDates
find	Used to loop through all the references in the invoice file to see if there is a match	Int	Local	IDates
result	Stores the value of the confirmation of whether or not it is the correct invoice to change	Int	Local	IDates
start	Whilever this variable is equal to zero the validation routine will run - it ensures that the start date entered is of the correct format	Int	Local	IDates
endrange	Whilever this variable is equal to zero the validation routine will run - it ensures that the end date entered is after the start date	Int	Local	IDates
end	Whilever this variable is equal to zero the validation routine will run - it ensures that the end date entered is of the correct format.	Int	Local	IDates
looking	Invoice reference entered by the user to have the dates changed	Char	Local	IPay
compare	Returns a value of whether there is a match between the invoice reference entered and one stored in the file	Int	Local	IPay
find	Used to loop through all the references in the invoice file to see if there is a match	Int	Local	IPay
result	Stores the value of the confirmation of whether or not it is the correct invoice to change	Int	Local	IPay

payment	Whilever this variable is equal to zero the validation routine will run - it ensures that the payment status is either 0 or 1	Int	Local	IPay
quotenum	Quote reference is passed as a parameter to find information to output to the screen	Char	Local	LocateQuoteCheck
find	Used to loop through all the references in the quote file to see if there is a match	Int	Local	LocateQuoteCheck
compare	Returns a value of whether there is a match between the quote reference stored in the invoice file and one stored in the quotes file	Int	Local	LocateQuoteCheck
find	Used to loop through all the references in the invoice file to see if there is a match	Int	Local	DeleteInvoice
compare	Returns a value of whether there is a match between the invoice reference entered and one stored in the file	Int	Local	DeleteInvoice
looking	Invoice reference entered by the user to be deleted	Char	Local	DeleteInvoice
result	Stores the value of the confirmation of whether or not it is the correct invoice to change	Int	Local	DeleteInvoice
del	Used to loop through the file and move all positions by -1	Int	Local	DeleteInvoice
viewstatus	Controls while loop for outputting invoice view menu	Int	Local	ViewInvoice
viewinvoicechoice	User input for menu option on invoice view menu	Int	Local	ViewInvoiceMenu
looking	Invoice reference entered by the user to be viewed	Char	Local	IRef
compare	Returns a value of whether there is a match between the invoice reference entered and one stored in the file	Int	Local	IRef
find	Used to loop through all the references in the invoice file to see if there is a match	Int	Local	IRef
looking	Customer reference entered by the user to view an invoice	Char	Local	ICustRef
compare	Returns a value of whether there is a match between the customer reference entered and one stored in the file	Int	Local	ICustRef
find	Used to loop through all the references in the invoice file to see if there is a match	Int	Local	ICustRef
custnum	Customer reference is passed as a parameter to output customer information with an invocie	Char	Local	LocateCustInvoice
compare	Returns a value of whether the record space is in use or not	Int	Local	LocateCustInvoice
quotenum	Quote reference is passed as a parameter to output quote information with an invocie	Char	Local	LocateQuote
compare	Returns a value of whether there is a match between the quote reference entered and one stored in the file	Int	Local	LocateQuote
find	Used to loop through all the references in the quote file to see if there is a match	Int	Local	LocateQuote
compare	Returns a value of whether there is a match between the payment status of those in the file and the character '1' to find unpaid invoices	Int	Local	IUnpaid
find	Used to loop through all the references in the quote file to see if there is a match	Int	Local	IUnpaid
count	Used to loop through all variables stored in the invoice file	Int	Local	ReadBackInvoiceFile
count	Used to loop through all variables stored in the invoice file	Int	Local	ReWriteInvoiceFile
stockref	Unique identifier for each item of stock	Char	Global	AddStock, UniqueStock, StPrice, StQuantity, DeleteStock, StID, StSortQ, ViewLowStock, ReadBackStockFile, ReWriteStockFile
quantity	The amount of each item of stock	Char	Global	AddStock, StPrice, StQuantity, DeleteStock, StID, StSortQ, ViewLowStock, ReadBackStockFile, ReWriteStockFile

colour	Colour of stock item such as paint	Char	Global	AddStock, StPrice, StQuantity, DeleteStock, StID, PrintArray, ReadBackStockFile, ReWriteStockFile
volume	Volume of paint	Char	Global	AddStock, DeleteStock, StID, ReadBackStockFile, ReWriteStockFile
type	Type of paint	Char	Global	AddStock, StPrice, StQuantity, DeleteStock, StID, PrintArray, ReadBackStockFile, ReWriteStockFile
stockprice	Price of 1 item of stock	Char	Global	FindStock, AddStock, StPrice, StQuantity, DeleteStock, StID, ReadBackStockFile, ReWriteStockFile
nsti	Keeps track of how many items of stock are stored in the file at one time	Int	Global	FindStock, AddStock, UniqueStock, StPrice, StQuantity, DeleteStock, StID, StSortQ, ViewLowStock, ReadBackStockFile, ReWriteStockFile
nstc	Keeps track of how many items of stock are stored in the file at one time	Char	Global	AddStock, DeleteStock, ReadBackStockFile, ReWriteStockFile
stockstatus	Controls while loop for outputting stock menu	Int	Local	Stock
stockchoice	User input for menu option on stock menu	Int	Local	StockMenu
ref	Whilever this variable is equal to zero the validation routine will run - it ensures that the stock reference entered is unique	Int	Local	AddStock
col	Whilever this variable is equal to zero the validation routine will run - it ensures that data has been entered	Int	Local	AddStock
ty	Whilever this variable is equal to zero the validation routine will run - it ensures that data has been entered	Int	Local	AddStock
price	Whilever this variable is equal to zero the validation routine will run - it ensures that the price entered is within a sensible range	Int	Local	AddStock
vol	Whilever this variable is equal to zero the validation routine will run - it ensures that the volume entered is within a sensible range	Int	Local	AddStock
quant	Whilever this variable is equal to zero the validation routine will run - it ensures that the volume entered is within a sensible range	Int	Local	AddStock
stock	Stock reference is passed as a parameter to check that the reference entered is unique	Char	Local	UniqueStock
uniqueref	Returns a value back to the original function to say whether or not the reference is unique	Int	Local	UniqueStock
find	Used to loop through all the references in the stock file to see if there is a match	Int	Local	UniqueStock
compare	Returns a value of whether there is a match between the stock reference entered and one stored in the file	Int	Local	UniqueStock
len	Stores the number of characters entered from the input to check data has been entered	Int	Local	UniqueStock
presence	Colour is passed as a parameter to check that data has been entered	Char	Local	PresColour
length	Stores the number of characters entered from the input to check data has been entered	Int	Local	PresColour
presence	Type is passed as a parameter to check that data has been entered	Char	Local	PresType
length	Stores the number of characters entered from the input to check data has been entered	Int	Local	PresType
rprice	Price of stock is passed as a parameter to check that data entered is within a suitable range	Char	Local	RangePrice
valid	Returns a value back to the original function to say whether or not the price is within the range	Int	Local	RangePrice
intprice	Converts price passed as paramter to integer to do the range check	Int	Local	RangePrice
len	Stores the number of characters entered from the input to check data has been entered	Int	Local	RangePrice
vol	Volume is passed as a parameter to check that data entered is within a suitable range	Char	Local	RangeVolume
valid	Returns a value back to the original function to say whether or not the volume is within the range	Int	Local	RangeVolume

intvol	Converts volume passed as paramter to integer to do the range check	Int	Local	RangeVolume
len	Stores the number of characters entered from the input to check data has been entered	Int	Local	RangeVolume
quant	Quantity is passed as a parameter to check that data entered is within a suitable range	Char	Local	RangeQuantity
valid	Returns a value back to the original function to say whether or not the quantity is within the range	Int	Local	RangeQuantity
intquant	Converts quantity passed as paramter to integer to do the range check	Int	Local	RangeQuantity
len	Stores the number of characters entered from the input to check data has been entered	Int	Local	RangeQuantity
changestatus	Controls while loop for outputting change stock menu	Int	Local	ChangeStock
changestockchoice	User input for menu option on change stock menu	Int	Local	ChangeStockMenu
looking	Stock reference entered by the user	Char	Local	StPrice
compare	Returns a value of whether there is a match between the stock reference entered and one stored in the file	Int	Local	StPrice
find	Used to loop through all the references in the stock file to see if there is a match	Int	Local	StPrice
result	Stores the value of the confirmation of whether or not it is the correct item of stock to be changed	Int	Local	StPrice
price	Whilever this variable is equal to zero the validation routine will run - it ensures that the price entered is within a sensible range	Int	Local	StPrice
looking	Stock reference entered by the user	Char	Local	StQuantity
compare	Returns a value of whether there is a match between the stock reference entered and one stored in the file	Int	Local	StQuantity
find	Used to loop through all the references in the stock file to see if there is a match	Int	Local	StQuantity
result	Stores the value of the confirmation of whether or not it is the correct item of stock to be changed	Int	Local	StQuantity
quant	Whilever this variable is equal to zero the validation routine will run - it ensures that the price entered is within a sensible range	Int	Local	StQuantity
find	Used to loop through all the references in the stock file to see if there is a match	Int	Local	DeleteStock
compare	Returns a value of whether there is a match between the stock reference entered and one stored in the file	Int	Local	DeleteStock
del	Used to loop through the file and move all positions by -1	Int	Local	DeleteStock
looking	Stock reference entered by the user	Char	Local	DeleteStock
result	Stores the value of the confirmation of whether or not it is the correct item of stock to be deleted	Int	Local	DeleteStock
viewstatus	Controls while loop for outputting view stock menu	Int	Local	ViewStock
viewstockchoice	User input for menu option on view stock menu	Int	Local	ViewStockMenu
looking	Stock reference entered by the user	Char	Local	StID
compare	Returns a value of whether there is a match between the stock reference entered and one stored in the file	Int	Local	StID
find	Used to loop through all the references in the stock file to see if there is a match	Int	Local	StID
sref	Copies the stock reference from the file to this variable	Char	Local	StSortQ
intquant	Extracts quantity and converts it to an integer	Int	Local	StSortQ
tempquant	Copies the quantity of the item of stock to this variable	Char	Local	StSortQ
count	Loops through all the items of stock store in the file	Int	Local	StSortQ
n	Number of items of stock stored in the file passed as a parameter	Int	Local	BubbleSort
sref	Stock reference for each item stored passed as a parameter	Char	Local	BubbleSort

intquant	Quantity of each item stored passed as a parameter	Int	Local	BubbleSort
position	Used to sort through every item of stock	Int	Local	BubbleSort
temp	Temporarily stores stock reference when moving in a bubble sort	Char	Local	BubbleSort
n	Number of items of stock stored in the file passed as a parameter	Int	Local	PrintArray
sref	Stock reference for each item stored passed as a parameter	Char	Local	PrintArray
intquant	Quantity of each item stored passed as a parameter	Int	Local	PrintArray
position	Used to loop through all items of stock in the file and output information in sorted order	Int	Local	PrintArray
find	Used to loop through all the references in the stock file to see if there is a match	Int	Local	ViewLowStock
quant	Stores the quantity of each item of stock in the file as an integer to check if the quantity is below two or not	Int	Local	ViewLowStock
count	Used to loop through all variables stored in the stock file	Int	Local	ReadBackStockFile
count	Used to loop through all variables stored in the stock file	Int	Local	ReWriteStockFile
staff	Stores staff reference as an integer	Int	Global	AddBooking,ChangeBooking,DeleteBooking,ClearSchedule,Staff1,Staff2,TodaysWork,ReadBackSchedule,ReWriteSchedule
date	Stores date of booking from 1-366	Int	Global	AddBooking,ChangeBooking,DeleteBooking,ClearSchedule,Staff1,Staff2,TodaysWork,ReadBackSchedule,ReWriteSchedule
hour	Stores hour of booking	Int	Global	AddBooking,ChangeBooking,DeleteBooking,ClearSchedule,Staff1,Staff2,TodaysWork,ReadBackSchedule,ReWriteSchedule
quoteno	Stores quote reference as an integer	Int	Global	AddBooking
booking	Stores quote reference to the schedule	Char	Global	AddBooking,ChangeBooking
addref	Stores quote reference	Char	Global	AddBooking
schedulestatus	Controls while loop for outputting schedule menu	Int	Local	Schedule
schedulechoice	User input for menu option on schedule menu	Int	Local	ScheduleMenu
staffref	Whilever this variable is equal to zero the validation routine will run - it ensures that the staff reference entered exsits within the system	Int	Local	AddBooking
qref	Whilever this variable is equal to zero the validation routine will run - it ensures the quote reference entered exists within the system	Int	Local	AddBooking
dateval	Whilever this variable is equal to zero the validation routine will run - it ensures that the date entered is in the correct format	Int	Local	AddBooking
datesyst	Whilever this variable is equal to zero the validation routine will run - it ensures that the date entered is in the future	Int	Local	AddBooking
starttime	Whilever this variable is equal to zero the validation routine will run - it ensures that the start time entered is within a certainrange	Int	Local	AddBooking
endtime	Whilever this variable is equal to zero the validation routine will run - it ensures that the end time entered is within a certain range	Int	Local	AddBooking
bookingdate	Stores the date of the booking entered by the user	Char	Local	AddBooking
endhour	Used to store the finsihing hour for that day to be used to store the booking at the correct time	Int	Local	AddBooking
ref	Staff reference is passed as a parameter to check that the staff reference exists within the system	Int	Local	StaffRefCheck
stafffound	Returns a value back to the original function to say whether or not the staff reference exists	Int	Local	StaffRefCheck
find	Used to loop through all the references in the staff file to see if there is a match	Int	Local	StaffRefCheck

compare	Returns a value of whether there is a match between the staff reference entered and one stored in the file	Int	Local	StaffRefCheck
reference	Stores the staff reference entered as a character for comparing with those in the file	Char	Local	StaffRefCheck
len	Stores the number of characters entered from the input to check data has been entered	Int	Local	StaffRefCheck
quoter	Quote reference is passed as a parameter to check that the quote reference exists within the system	Int	Local	QuoteRefCheckBooking
uniqueref	Returns a value back to the original function to say whether or not the quote reference exists	Int	Local	QuoteRefCheckBooking
find	Used to loop through all the references in the quote file to see if there is a match	Int	Local	QuoteRefCheckBooking
compare	Returns a value of whether there is a match between the quote reference entered and one stored in the file	Int	Local	QuoteRefCheckBooking
quote	Stores the quote reference entered as a character for comparing with those in the file	Char	Local	QuoteRefCheckBooking
date	Date of booking entered by the user is passed as a parameter to ensure it is of the correct format	Char	Local	DateValBooking
valid	Returns a value back to the original function to say whether or not the date is in the correct format	Int	Local	DateValBooking
months	Extracts the month entered from the date for validation	Int	Local	DateValBooking
days	Extracts the day from the date entered for validation	Int	Local	DateValBooking
years	Extracts the year from the date entered for validation	Float	Local	DateValBooking
len	Stores the number of characters entered from the input to check data has been entered	Int	Local	DateValBooking
date	Date of booking entered by the user is passed as a parameter to ensure it is in the future	Char	Local	SystemsClockBooking
month	Extracts the month entered from the date for validation	Int	Local	SystemsClockBooking
day	Extracts the day from the date entered for validation	Int	Local	SystemsClockBooking
year	Extracts the year from the date entered for validation	Int	Local	SystemsClockBooking
valid	Returns a value back to the original function to say whether or not the date is in the future	Int	Local	SystemsClockBooking
stime	Start time is passed as a parameter to ensure it is within a certain range	Int	Local	StartRange
valid	Returns a value back to the original function to say whether or not the start time is in the correct range	Int	Local	StartRange
etime	End time is passed as a parameter to ensure it is within a certain range	Int	Local	EndRange
valid	Returns a value back to the original function to say whether or not the end time is in the correct range	Int	Local	EndRange
quotenum	Quote reference entered by the user	Int	Local	ChangeBooking
ref	Stores quote reference	Char	Local	ChangeBooking
newdate	Stores the result of the date entered as a day from 1-366	Int	Local	ChangeBooking
bookingdate	Stores old date of booking entered by the user	Char	Local	ChangeBooking
newbookingdate	Stores new date of booking entered by the user	Char	Local	ChangeBooking
endhour	Used to store the finsihing hour for that day to be used to store the booking at the correct time	Int	Local	ChangeBooking
staffref	Whilever this variable is equal to zero the validation routine will run - it ensures that the staff reference entered exsits within the system	Int	Local	ChangeBooking
datevalold	Whilever this variable is equal to zero the validation routine will run - it ensures that the date entered is in the correct format	Int	Local	ChangeBooking
datevalnew	Whilever this variable is equal to zero the validation routine will run - it ensures that the date entered is in the correct format	Int	Local	ChangeBooking

starttime	Whilever this variable is equal to zero the validation routine will run - it ensures that the start time entered is within a certainrange	Int	Local	ChangeBooking
endtime	Whilever this variable is equal to zero the validation routine will run - it ensures that the end time entered is within a certain range	Int	Local	ChangeBooking
qref	Whilever this variable is equal to zero the validation routine will run - it ensures the quote reference entered exists within the system	Int	Local	ChangeBooking
bookingdate	Stores date of booking entered by the user	Char	Local	DeleteBooking
endhour	Used to store the finsihing hour for that day to be used to delete the booking at the correct times	Int	Local	DeleteBooking
staffref	Whilever this variable is equal to zero the validation routine will run - it ensures that the staff reference entered exsits within the system	Int	Local	DeleteBooking
dateval	Whilever this variable is equal to zero the validation routine will run - it ensures that the date entered is in the correct format	Int	Local	DeleteBooking
starttime	Whilever this variable is equal to zero the validation routine will run - it ensures that the start time entered is within a certainrange	Int	Local	DeleteBooking
endtime	Whilever this variable is equal to zero the validation routine will run - it ensures that the end time entered is within a certain range	Int	Local	DeleteBooking
viewstatus	Controls while loop for outputting view schedule menu	Int	Local	ViewSchedule
viewschedulechoice	User input for menu option on view schedule menu	Int	Local	ViewScheduleMenu
leap	Stores a value which determine whether or not the year is a leap year	Int	Local	DateOutput
dateofyear	Date from 1-366 is passed as a paramter to convert into the format DD/MM/YYYY	Int	Local	DateOutput
leap	Stores a value which determine whether or not the year is a leap year	Int	Local	TodaysWork
value	Stores the result of the date entered as a day from 1-366	Int	Local	TodaysWork
date	Stores the result of value -1 for the schedule	Int	Local	TodaysWork
tempquote	Stores quote reference	Char	Local	TodaysWork
charref	Quote reference is passed as a parameter to find information on the quote to output with the work for that day	Char	Local	LocateQuoteWork
find	Used to loop through all the references in the quote file to see if there is a match	Int	Local	LocateQuoteWork
compare	Returns a value of whether there is a match between the quote reference entered and one stored in the file	Int	Local	LocateQuoteWork
custno	Customer reference is passed as a parameter to find information on the customer to ouptut with the work for that day	Char	Local	LocateCustSchedule
compare	Returns a value of whether the record space is in use or not	Int	Local	LocateCustSchedule
datein	Date is passed as a parameter to be converted into a day from 1-366	Char	Local	ConvertDate
year	Extracts the year entered from the date for validation	Float	Local	ConvertDate
month	Extracts the month from the date entered for validation	Int	Local	ConvertDate
day	Extracts the dayfrom the date entered for validation	Int	Local	ConvertDate
value	Stores the result of the date entered as a day from 1-366	Int	Local	ConvertDate
valid	Stores a value which determine whether or not the year is a leap year	Int	Local	ConvertDate
newdatein	Date is passed as a parameter to be converted into a day from 1-366	Char	Local	ConvertDateNew
year	Extracts the year entered from the date for validation	Float	Local	ConvertDateNew
month	Extracts the month from the date entered for validation	Int	Local	ConvertDateNew

day	Extracts the dayfrom the date entered for validation	Int	Local	ConvertDateNew
value	Stores the result of the date entered as a day from 1-366	Int	Local	ConvertDateNew
valid	Stores a value which determine whether or not the year is a leap year	Int	Local	ConvertDateNew
linksquoteref	Stores the quote reference	Char	Global	AddQuote,ReadBackLinksFile,ReWriteLinksFile
linksstockref	Stores the stock reference	Char	Global	AddQuote,ReadBackLinksFile,ReWriteLinksFile
linksquantity	Stores the quantity required for a specific item of stock	Char	Global	FindStock,ReadBackLinksFile,ReWriteLinksFile
nli	Stores number of links	Int	Global	AddQuote,ReadBackLinksFile,ReWriteLinksFile
nlc	Stores number of links	Char	Global	ReadBackLinksFile,ReWriteLinksFile

```
/*****************************************************************************
Program: SoftwareDev
File: SoftwareDev.cpp
Functions: main, AddBooking, AddCustomer, AddInvoice, AddQuote, AddStock, AddStockQuote, Backup, BubbleSort, Buffer ChangeBooking, ChangeCus
ChangeCustomerMenu, ChangeInvoice, ChangeInvoiceMenu, ChangeQuote, ChangeQuoteMenu, ChangeStaff, ChangeStaffMenu, ChangeStock,
ChangeStockMenu, ChgLoa, ChgPassword, ChgUsername, CHomeAdd, ClearSchedule, CName,ConvertDate,ConvertDateNew CRef, CTelNum, CustomerMenu, Cu
DateOutput, DateVal,DateValBooking,DateValEnd,DateValInvoice,DateValStart,
DeleteBooking, DeleteCustomer, DeleteInvoice, DeleteQuote, DeleteStaff, DeleteStock,EndRange,FindLoa, FindStock, ICustRef, IDates, InvoiceMe
Invoices, IPay, IRef, IUnpaid, LocateCust, LocateCustInvoice, LocateCustIVal,LocateCustQVal,LocateCustSchedule,LocateQuote,LocateQuoteCheck,
Login, MainMenu, Menu, PassLoginCheck, PasswordVal, PostCodeVal, PostCodeValCust, PresColour, PresType, PresVal, PresValFirst,
PresValJ, PresValL, PresValLast, PresValO, PresValOne,PresValT,PresValTh,PresValThree,PresValTitle,PresValTr,PresValT, PrintArray, QCustRef,
QDay, QPrice, QRef, QuoteMenu, QuotePriceRange, QuoteRefCheck, QuoteRefCheckBooking, Quotes, RangeCheck, RangeDate, RangeDay, RangePaid, Ran
RangeVolume, ReadBackInvoiceFile, ReadBackLinksFile, ReadBackQuotesFile, ReadBackScheduleFile, ReadBackStaffFile, ReadBackStockFile, Recover
ReWriteInvoiceFile, ReWriteLinksFile, ReWriteQuotesFile, ReWriteScheduleFile, ReWriteStaffFile, ReWriteStockFile, Schedule, ScheduleCheck,
ScheduleMenu, SHomeAdd, Staff, Staff1, Staff2, StaffMenu, StaffRefCheck, StartRange, STelNum, StID, Stock, StockCalc, StockMenu, StockVal, S
SystemsClockBooking, SystemsClockValStart, TelVal, TelValCust, TelValE, TodaysWork, UniqueInvoice, UnwiueQ, UniqueStaff, UniqueStock, Unique
ViewCustomer, ViewCustomerMenu, ViewInvoice, ViewInvoiceMenu, ViewLowStock, ViewQuote, ViewQuoteMenu, ViewSchedule, ViewScheduleMenu, ViewSt
Description:
Author: Fion McReynolds
Environment: Borland C++ Pro 6.0
Notes:
Revisions: 06/10/2023
*****************************************************************************/
#include<string.h>
#include<fstream.h>
#include<stdlib.h>
#include<math.h>
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<vcl.h>
#include<time.h>
#pragma hdrstop

//---------------------------------------------------------------------------

#pragma argsused

int MainMenu();
int Buffer(char bufferpassed[30],int length);
// STAFF
int Staff();
int Login();
int WhileMenu();
int Menu();
int StaffMenu();
int AddStaff();

int ChangeStaff();
int ChangeStaffMenu();
int SHomeAdd();
int STelNum();
int ChgLoa();
int ChgUsername();
int ChgPassword();
```

```
int DeleteStaff();
int ViewStaff();
int Backup();
int Recovery();

int ReadBackStaffFile();
int ReWriteStaffFile();

char FileName1[80] = "StaffFileSD";
char staffref[10][3];
char fnamestaff[10][15];
char lnamestaff[10][15];
char oneadstaff[10][30];
char twoadstaff[10][30];
char threeadstaff[10][30];
char pcodestaff[10][9];
char telnostaff[10][12];
char emtel[10][12];
char ninum[10][15];
char username[10][15];
char password[10][15];
char loa[10][2];
int nsi;
char nsc[3];
int isfirsttime=1;
int level;

int UniqueStaff(char staffbuffer[2]);
int PresVal(char fnamestaff[15]);
int PresValL(char lnamestaff[15]);
int PresValO(char oneadstaff[30]);
int PresValT(char twoadstaff[30]);
int PresValTh(char threeadstaff[30]);
int PostCodeVal(char pcodestaff[9]);
int TelVal(char telnostaff[12]);
int TelValE(char emtel[12]);
int ValidNI(char ninum[15]);
int UniqueUser(char user[15]);
int PasswordVal(char password[15]);
int RangeCheck(char loa[2]);
int UserLoginCheck(char username[15]);
int PassLoginCheck(char password[15]);
int FindLoa(char username[15]);

// CUSTOMERS
int Customers();
int CustomerMenu();
int AddCustomer();

int ChangeCustomer();
int ChangeCustomerMenu();
int CHomeAdd();
int CTelNum();
```

```
int DeleteCustomer();

int ViewCustomer();
int ViewCustomerMenu();
int CName();
int CRef();

char FileName2[80] = "CustomerFileSD";

typedef struct tag_cr{
        char title[10];
        char fnamecust[15];
        char lnamecust[15];
        char oneadcust[30];
        char twoadcust[30];
        char threeadcust[15];
        char pcodecust[9];
        char telnocust[12];
        char flag[2];
}CUSTOMER_RECORD;
int custref;
CUSTOMER_RECORD a_cust;

int CustRefVal(char tempcust[3]);
int PresValTitle(char title[10]);
int PresValFirst(char fnamecust[15]);
int PresValLast(char lnamecust[15]);
int PresValOne(char oneadcust[30]);
int PresValTwo(char twoadcust[30]);
int PresValThree(char threeadcust[15]);
int PostcodeValCust(char pcodecust[9]);
int TelValCust(char telnocust[12]);
int ScheduleCheck(int custef);

// QUOTES
int Quotes();
int QuoteMenu();
int AddQuote();

int ChangeQuote();
int ChangeQuoteMenu();
int QPrice();
int QDay();

int DeleteQuote();

int ViewQuote();
int ViewQuoteMenu();
int QRef();
int LocateCust(char custno[3]);
int QCustRef();

int StockCalc();
int FindStock(char lstock[3], int count);
```

```
char FileName4[80] = "QuotesFileSD";
char quoteref[10][3];
char custno[10][3];
char quotedate[10][11];
char mainjobdesc[10][50];
char numofdays[10][3];
char labourq[10][5];
char mileage[10][4];
char vat[10][5];
char stockcost[10][5];
char totalcost[10][5];
int AddStockQuote(char quoteref[3]);

int nqi;
char nqc[3];

int ReadBackQuotesFile();
int ReWriteQuotesFile();

int numofitems;
int stock=0;
int cost =0;

int UniqueQ(char quoteref[4]);
int LocateCustQVal(char custno[3]);
int DateVal(char quotedate[11]);
int PresValJ(char mainjobdesc[50]);
int PresValTr(char mileage[4]);
int RangeDay(char numofdays[3]);
int QuotePriceRange(char cost[5]);

char temptime[40];
int yeart;
char monthchar[4];
int dayst;
int rawtime;
int monthval;
int monthint;

// INVOICES
int Invoices();
int InvoiceMenu();
int AddInvoice();

int ChangeInvoice();
int ChangeInvoiceMenu();
int IDates();
int IPay();

int DeleteInvoice();

int ViewInvoice();
int ViewInvoiceMenu();
int IRef();
```

```
int ICustRef();
int IUnpaid();
int LocateCustInvoice(char custnum[3]);
int LocateQuote(char quotenum[3]);
int LocateQuoteCheck(char quotenum[3]);

char FileName5[80] = "InvoiceFileSD";
char invoiceref[10][3];
char custnum[10][3];
char quotenum[10][3];
char invoicedate[10][11];
char jobstartdate[10][11];
char jobenddate[10][11];
char paid[10][2];
int nii;
char nic[3];

int monthstart;
int yearstart;
int daystart;

int ReadBackInvoiceFile();
int ReWriteInvoiceFile();

int UniqueInvoice(char invoiceref[3]);
int LocateCustIVal(char custnum[3]);
int DateValInvoice(char invoicedate[11]);
int DateValStart(char jobstartdate[11]);
int SystemsClockValStart(char jobstartdate[11]);
int DateValEnd(char jobenddate[11]);
int RangeDate(char jobenddate[11]);
int RangePaid(char paid[2]);
int QuoteRefCheck(char quotenum[3]);

// STOCK
int Stock();
int StockMenu();
int AddStock();

int ChangeStock();
int ChangeStockMenu();
int StPrice();
int StQuantity();

int DeleteStock();

int ViewStock();
int ViewStockMenu();
int StID();
int StSortQ();

int ViewLowStock();
int ReadBackStockFile();
int ReWriteStockFile();
```

```
char FileName3[80] = "StockFileSD";
char stockref[10][10];
char quantity[10][10];
char colour[10][15];
char volume[10][4];
char type[10][10];
char stockprice[10][8];
int nsti;
char nstc[3];

void BubbleSort(char sref[][10],int intquant[],int n);
void PrintArray(char sref[][10],int intquant[],int n);

int UniqueStock(char stockref[10]);
int RangeQuantity(char quantity[10]);
int PresColour(char colour[15]);
int RangeVolume(char volume[4]);
int PresType(char type[10]);
int RangePrice(char stockprice[8]);
int StockVal(char stockref[10]);
int StockInformation(char stockref[10]);

// SCHEDULE
int Schedule();

int ScheduleMenu();
int AddBooking();
int ConvertDate(char datein[12]);
int ConvertDateNew(char newdatein[12]);
int ChangeBooking();
int DeleteBooking();
int ClearSchedule();



int ViewSchedule();
int DateOutput(int dateofbooking);
int ViewScheduleMenu();
int TodaysWork();
int Staff1();
int Staff2();


int ReadBackScheduleFile();
int ReWriteScheduleFile();


int StaffRefCheck(int ref);
int QuoteRefCheckBooking(int quotenum);
int DateValBooking(char date[12]);
int SystemsClockBooking(char date[12]);
int StartRange(int stime);
int EndRange(int etime);
int LocateQuoteWork(char charref[3]);
int LocateCustSchedule(char custno[3]);
```

```
int staff;
int date;
int hour;
int quoteno;
char booking[3][370][14][3];  //staff week date hour
char addref[3];


char FileName7[80] = "ScheduleFileSD";

// Links File
char FileName6[80] = "LinksFile";
char linksquoteref[10][3];
char linksstockref[10][3];
char linksquantity[10][4];
int nli;
char nlc[3];

int ReadBackLinksFile();
int ReWriteLinksFile();


int main(int argc, char* argv[])
{
int loginstatus;
//While loop which allows the user to see the first output to the screen in the Login() function
while(loginstatus != 2)
        {
        loginstatus = Login();
        }
getch();
return 0;
}

//Routine that either allows a user to enter their username and password to access the system or if its an empty file allows a new staff mem
int Login()
{
//Outputted to the screen - first thing a user will see
cout<<"Welcome";
cout<<"\n*******";
cout<<"\n";
cout<<"\nPress enter to continue.";
getch();
clrscr();

ReadBackStaffFile();
cin.get();

// checking that the number of users isn't 0
 //If number of users isnt 0, sets isfirsttime = 0 as it is not the first user on the system it then takes user to WhileMenu() which will al
if(nsi != 0)
        {
        isfirsttime = 0;
        WhileMenu();
        } //endif
```

```
 //If number of users is =0, sets isfirsttime=1 to show it is the first user on the system, it then tkes the user to the WhileMenu() where a
if(nsi==0)
        {
        WhileMenu();
        isfirsttime=1;
        }
return 0;
}

//A routine which allows a user to gain access to the system
int WhileMenu()
{
//local variables
int insg=0;
int ref=0;
int fname=0;
int lname=0;
int adone=0;
int adthree=0;
int pcode=0;
int tel=0;
int emertel=0;
int uquser=0;
int passvalone=0;
int userval=0;
int passvaltwo=0;

char staffin[30];
int staffbuff=0;
char telin[30];
int telbuff=0;
char emtelin[30];
int emtelbuff=0;
char ninumin[30];
int nibuff=0;
if(isfirsttime==1)
        {
        ReadBackStaffFile();
        cin.get();
        //Outputs welcome message to the scren
        cout<<"\n Welcome to Stuart Watson's Painting and Decorating";
        cout<<"\n************************************************";
        cout<<"\n You are the first user to access the system!";
        cout<<"\n";
        cout<<"\nPlease enter the following information as prompted below.";
        cout<<"\n";
        //User enters all data to be saved regarding a member of staff - all validated upon entry
        while(staffbuff==0 || ref==0)   //While loop that controls the validation to ensure that the staff reference isnt in use
                {
                cout<<"\nEnter staff reference: ";
                cin.getline(staffin,30);
                staffbuff = Buffer(staffin,1);
                if(staffbuff!=0 && ref==0)
                        {
```

```
                ref = UniqueStaff(staffin);
                }//endif
        }// end while
    strcpy(staffref[nsi],staffin);

    while(fname==0)  //While loop that controls the validation to ensure that a first name is entered
        {
        cout<<"\nEnter first name: ";
        cin.getline(fnamestaff[nsi],15);
        fname = PresVal(fnamestaff[nsi]);
        }// end while

    while(lname==0)   //While loop that controls the validation to ensure that a last name is entered
        {
        cout<<"\nEnter last name: ";
        cin.getline(lnamestaff[nsi],15);
        cout<<"\n";
        lname = PresValL(lnamestaff[nsi]);
        }// end while

    while(adone == 0)   //While loop that controls the validation to ensure that address line 1 is entered
        {
        cout<<"\nEnter address line 1: ";
        cin.getline(oneadstaff[nsi],30);
        adone = PresValO(oneadstaff[nsi]);
        }//end while

cout<<"\nEnter address line 2: "; //No validation as address line2 is not mandatory
cin.getline(twoadstaff[nsi],30);

    while(adthree == 0)  //While loop that controls the validation to ensure that address line 3 is entered
        {
        cout<<"\nEnter address line 3: ";
        cin.getline(threeadstaff[nsi],30);
        adthree = PresValTh(threeadstaff[nsi]);
        }// end while

    while(pcode==0)  //While loop that controls the validation to ensure that the postcode entered is a valid UK postcode
        {
        cout<<"\nEnter postcode: ";
        cin.getline(pcodestaff[nsi],9);
        cout<<"\n";
        pcode = PostCodeVal(pcodestaff[nsi]);
        }//endwhile

    while(telbuff==0 || tel==0) //While loop that controls the validation to ensure that the mobile number entered is in the correct for
        {
        cout<<"\nEnter mobile number: ";
        cin.getline(telin,30);
        telbuff = Buffer(telin,11);
        if(telbuff!=0 && tel==0)
                {
                tel = TelVal(telin);
                }//endif
        }//end while
```

```
        strcpy(telnostaff[nsi],telin);
        while(emtelbuff==0 || emertel == 0) //While loop that controls the validation to ensure that the mobile number entered is in the cor
                {
                cout<<"\nEnter emergency mobile number: ";
                cin.getline(emtelin,30);
                emtelbuff = Buffer(emtelin,11);
                if(emtelbuff!=0 && emertel==0)
                        {
                        emertel = TelValE(emtelin);
                        }//endif
                }// end while
        strcpy(emtel[nsi],emtelin);
        while(nibuff==0 || insg == 0) //while loop that controls validation to ensure the national insurance is of the correct format
                {
                cout<<"\nEnter national insurance number (FORMAT:AB 12 34 56 C): ";
                cin.getline(ninumin,30);
                nibuff = Buffer(ninumin,13);
                if(nibuff!=0 && insg==0)
                        {
                        insg = ValidNI(ninumin);
                        }//endif
                }// end while
        strcpy(ninum[nsi],ninumin);
        while(uquser == 0)  //while loop that controls validation to ensure the username entered is unqiue
                {
                cout<<"\nEnter username: ";
                cin.getline(username[nsi],15);
                uquser = UniqueUser(username[nsi]);
                }// end while
        while(passvalone==0)  //while loop that controls validation to ensure the password entered conforms to a set of rules
                {
                cout<<"\nEnter password (Minimum 8 characters with at least one piece of punctuation): ";
                cin.getline(password[nsi],15);
                passvalone = PasswordVal(password[nsi]);
                }// end while

      strcpy(loa[nsi],"3");  //Copies the level of access to the highest priority as the first person to use the system will have administra
        nsi = nsi +1;         //Increases the number of staff members stored in the file by one
        itoa(nsi,nsc,10);
        ReWriteStaffFile();   // Data is added to the staff file
        }//end if
clrscr();
//If there are 1 or more users stored in the staff file user enters their username and password
if(isfirsttime==0)
        {
        clrscr();
        ReadBackStaffFile();
        while(userval==0)
                {
                cout<<"\nEnter username: ";    //Checks that the username exists within in the file and if it doesn't, doesn't allow the use
                cin.getline(username[nsi],15);
                userval = UserLoginCheck(username[nsi]);
                }
        while(passvaltwo==0)
                {
```

```
                cout<<"\nEnter password: ";      //Checks that the password exists within in the file and if it doesn't, doesn't allow the u
                cin.getline(password[nsi],15);
                passvaltwo = PassLoginCheck(password[nsi]);
                }
        if(userval==1&&passvaltwo==1)
                {                       //Only allows access to the Main Menu if both username and password are correct
                Menu();
                }//end if
        }//end if
getch();
return 0;
}

//This function determines whether or not the main menu should be outputted to the screen
int Menu()
{
clrscr();
int mainstatus=0;
//While loop that outputs the main menu
while(mainstatus != 7)
        {
        mainstatus = MainMenu();
        }//endwhile
getch();
return 0;
}

//Checks that the username entered is correct
int UserLoginCheck(char user[15])
{
//local variables
int unique=0;
int find;
int compare;


ReadBackStaffFile();
for (find = 0; find < nsi; find++)      //searches through all the staff in the staff file
        {
        compare = strcmpi(username[find], user);   //Compares each staff members username in the file with the one entered
        if (compare == 0)  //If compare=0 then the username entered is the same as one stored in the file
                {
                //Username has been found
                unique = 1;
                FindLoa(username[find]);   //Routine that finds the logged in users level of access for use throughout the system
                return unique;
                } //endif found record
        }//endfor
if(compare!=0)
        {
        //Error message is outputted if no username matches
        cout<<"\nUsername not found. Please try again.";
        getch();
        }//end else
return unique;
```

```
}

//Checks that the password entered are correct
int PassLoginCheck(char pass[15])
{
//local variables
int unique=0;
int find;
int compare;

ReadBackStaffFile();
for (find = 0; find < nsi; find++)      //searches through all the staff in the staff file
        {
        compare = strcmpi(password[find], pass);   //Compares each staff members password in the file with the one entered
        if (compare == 0) //If compare=0 then the password entered is the same as one stored in the file
                {
                unique = 1;    //Password has been found
                return unique;
                } //endif found record
        }//endfor
if(compare!=0)
        {
        //Error message is outputted if no username matches
        cout<<"\nPassword not found. Please try again.";
        getch();
        }//end else
return unique;
}

//Finds the level of access for hierarchial use throughout the system
int FindLoa(char user[15])
{
//local variables
int find;
int compare;


for(find=0;find<nsi;find++)         //Searches through all the staff in the file
        {
        compare = strcmpi(user,username[find]);      //Compares each username in the file with the one entered
        if(compare==0)  //If compare=0 then the username passed is the same as one stored in the file
                {
                //Copies level of acces into the global variable 'level' for hierachial use through the system while they are logged in
                level = atoi(loa[find]);
                }//end if
        }//end for
getch();
return 0;
}


//Outputs the main menu to the screen allowing the user to access different parts of the system
int MainMenu()
{
```

```
system("Color 0C");  //Once entry is confirmed, text turns red
int choice;

//Outputs menu options to the users screen
cout<<"\n \t Main Menu";
cout<<"\n \t ********* \n \n";
cout<<"\n \t 1. Staff";
cout<<"\n \t 2. Customers";
cout<<"\n \t 3. Quotes";
cout<<"\n \t 4. Invoices";
cout<<"\n \t 5. Stock";
cout<<"\n \t 6. Schedule";
cout<<"\n \t 7. Quit";
cout<<"\n";
cout<<"\n \t Enter choice: ";
getch();
cin>>choice;   //Users choice from the menu is entered
clrscr();    //Clears main menu from the screen
switch(choice)  //Allows different functions to be run depening on the users input
        {
        case 1: //If choice ==1 the staff function can be run
             {
             if(level==3)     //Only allows access to the staff with level of acess of '3'
                {
                Staff();
                }
             else
                {
                //Error message if users level of access is not 3.
                cout<<"\n Not authorised";
                getch();
                }
             break;
             }
        case 2: //If choice ==2 the customer function can be run
             {
             Customers();
             break;
             }
                               //Entry into specific menus
        case 3: //If choice ==3 the quote function can be run
             {
             Quotes();
             break;
             }

        case 4: //If choice ==4 the invoice function can be run
             {
             Invoices();
             break;
             }
        case 5:  //If choice ==5 the stock function can be run
             {
             Stock();
             break;
```

```
        }
      case 6:   //If choice ==6 the schedule function can be run
        {
        Schedule();
        break;
        }
      case 7:   //If choice ==7 the option is quit so the program is closed
        {
        break;
        }
      default:
          {
          //If any other digit is typed either less than 1 or greater than 7 - error message is outputted
          cout<<"\nPlease enter a number between 1 and 7.";
          getch();
          break;
          }
      }//endcase
clrscr();
getch();
return choice;
}

//This function determines whether or not the staff menu should be outputted to the screen
int Staff()
{
int staffstatus=0;
//While menu that outputs Staff Menu
while(staffstatus != 7)
      {
      staffstatus = StaffMenu();
      }//endwhile
getch();
return 0;
}

//Outputs staff menu options to the user and allows them to choose one of them
int StaffMenu()
{
//Menu to access different aspects of the staff / admin information
int staffchoice;
//Ouputs staff options to the screen
cout<<"\n \t Staff Menu";
cout<<"\n \t **********\n \n";
cout<<"\n \t 1. Add Staff";
cout<<"\n \t 2. Change staff information";
cout<<"\n \t 3. Delete staff";
cout<<"\n \t 4. View staff";
cout<<"\n \t 5. Backup";
cout<<"\n \t 6. Recovery";
cout<<"\n \t 7. Return to Main Menu";
cout<<"\n";
cout<<"\n \t Enter choice: ";
getch();
cin>>staffchoice;   //Menu option entered by the user
```

```
clrscr();
switch(staffchoice)  //Allows different functions to be ran deoening on the menu option entered by the user
        {
        case 1:  //If staffchoice = 1 then the AddStaff function is ran
             {
             AddStaff();
             break;
             }
        case 2:  //If staffchoice = 2 then the ChangeStaff function is ran
             {
             ChangeStaff();
             break;                       //Depending on users input, routines for different parts of the menu
             }

        case 3: //If staffchoice = 3 then the DeleteStaff function is ran
             {
             DeleteStaff();
             break;
             }

        case 4: //If staffchoice = 4 then the ViewStaff function is ran
             {
             ViewStaff();
             break;
             }
        case 5: //If staffchoice = 5 then the Backup function is ran
             {
             Backup();
             break;
             }
        case 6:  //If staffchoice = 6 then the Recovery function is ran
             {
             Recovery();
             break;
             }
        case 7: //If staffchoice = 7 then the program is finished
             {
             break;
             }
        default:
               {
               //If an incorrect number is entered, error message is outputted
               cout<<"\nPlease enter a number between 1 and 7";
               getch();
               break;
               }
        }//endcase
clrscr();
getch();
return staffchoice;
}

//Adds new staff member to the staff file
int AddStaff()
{
```

```
//local variables
int insg=0;
int ref=0;
int fname=0;
int lname=0;
int adone=0;
int adthree=0;
int pcode=0;
int tel=0;
int emertel=0;
int uquser=0;
int passval=0;
int levrange=0;

int bufferstaff=0;
int telbuff=0;
int emtelbuff=0;
char emtelb[30];
char telb[30];
char staffb[30];
int nibuff=0;
char bni[30];

cout<<"\nAdd Staff";
cout<<"\n*********";

ReadBackStaffFile();
cin.get();

//User enters all data to be saved regarding a member of staff - all validated upon entry
while(bufferstaff==0 || ref==0)      //while loop that controls validation to ensure the staff reference is not already in use and a buffer
        {
        cout<<"\nEnter staff reference: ";
        cin.getline(staffb,30);
        bufferstaff=Buffer(staffb,1);
        if(bufferstaff!=0 && ref==0)
                {
                ref=UniqueStaff(staffb);
                }
        }//end while
strcpy(staffref[nsi],staffb);


while(fname==0)     //while loop that controls validation to ensure the first name is entered
        {
        cout<<"\nEnter first name: ";
        cin.getline(fnamestaff[nsi],15);
        fname = PresVal(fnamestaff[nsi]);
        }// end while

while(lname==0)     //while loop that controls validation to ensure the last name is entered
        {
        cout<<"\nEnter last name: ";
        cin.getline(lnamestaff[nsi],15);
        cout<<"\n";
```

```
        lname = PresValL(lnamestaff[nsi]);
        }// end while

while(adone == 0)                //while loop that controls validation to ensure that address line 1 is entered
        {
        cout<<"\nEnter address line 1: ";
        cin.getline(oneadstaff[nsi],30);
        adone = PresValO(oneadstaff[nsi]);
        }//end while

cout<<"\nEnter address line 2: ";    //Address line 2 is added - no validation as not all adresses have an address line 2
cin.getline(twoadstaff[nsi],30);


while(adthree == 0)           //while loop that controls validation to ensure that address line 3 is entered
        {
        cout<<"\nEnter address line 3: ";
        cin.getline(threeadstaff[nsi],30);
        adthree = PresValTh(threeadstaff[nsi]);
        }// end while

while(pcode==0)       //while loop that controls validation to ensure the postcode is of a correct uk format
        {
        cout<<"\nEnter postcode: ";
        cin.getline(pcodestaff[nsi],9);
        cout<<"\n";
        pcode = PostCodeVal(pcodestaff[nsi]);
        }//endwhile

while(telbuff==0||tel==0)    //while loop that controls validation to ensure the mobile is of the correct format: 07NNNNNNNN
        {
        cout<<"\nEnter mobile number: ";
        cin.getline(telb,30);
        telbuff=Buffer(telb,11);
        if(telbuff!=0 && tel==0)
                {
                tel = TelVal(telb);
                }
        }//end while

strcpy(telnostaff[nsi],telb);
while(emtelbuff==0 || emertel == 0)     //while loop that controls validation to ensure the mobile is of the correct format: 07NNNNNNNN
        {
        cout<<"\nEnter emergency mobile number: ";
        cin.getline(emtelb,30);
        emtelbuff=Buffer(emtelb,11);
        if(emtelbuff!=0 && emertel==0)
                {
                emertel = TelValE(emtelb);
                }
        }// end while
strcpy(emtel[nsi],emtelb);

while(nibuff==0 || insg == 0)     //while loop that controls validation to ensure the national insurance is of the correct format
        {
```

```
        cout<<"\nEnter national insurance number (FORMAT:AB 12 34 56 C): ";
        cin.getline(bni,30);
        cout<<"\n";
        nibuff = Buffer(bni,13);
        if(nibuff!=0 && insg==0)
                {
                insg = ValidNI(bni);
                }
        }// end while
strcpy(ninum[nsi],bni);

while(uquser == 0)     //while loop that controls validation to ensure the username is not already in use
        {
        cout<<"\nEnter username: ";
        cin.getline(username[nsi],15);
        uquser = UniqueUser(username[nsi]);
        }// end while
while(passval==0)                    //while loop that controls validation to ensure the password conforms to a set of rules
        {
        cout<<"\nEnter password (Minimum 8 characters with at least one piece of punctuation): ";
        cin.getline(password[nsi],15);
        passval = PasswordVal(password[nsi]);
        }// end while

while(levrange == 0)      //while loop that controls validation to ensure the access level is between1 and 3
        {
        cout<<"\nEnter level of access: ";
        cin.getline(loa[nsi],2);
        levrange = RangeCheck(loa[nsi]);
        }// end while

nsi = nsi +1;    //Increases the number of staff members stored in the file by one
itoa(nsi,nsc,10);

ReWriteStaffFile();   // Data is added to the staff file
getch();
return 0;
}


//Validation routine to check that the staff reference is unique and not already in use
int UniqueStaff(char staff[3])
{
//local variables
int uniqueref=1;
int find;
int compare;
int len;
int position;

len = strlen(staff); //Stores the length of the input


for(position=0;position<len;position++)     //loops through all characters in inout
        {
```

```
        if(isdigit(staff[position]))      //Checks to see if it an integer, if it is the rest of the loop runs
                {
                ReadBackStaffFile();
                for(find=0;find<nsi;find++)           // loops through all stored staff
                        {
                        compare = strcmpi(staffref[find], staff);      // compares each staff reference with the one enterd by the user
                        if (compare == 0)            // staff reference entered has been matched with one in the file
                                {
                                uniqueref = 0;
                                cout<<"\n Staff reference already in use. Please try again.";  //Error message if staff reference is already
                                getch() ;
                                } //endif
                        }//endfor
                }//end if
                else
                        {
                        uniqueref=0;
                        //Error message if the staff reference entered was not an integer.
                        cout<<"\nPlease ensure you have entered a number for the staff reference.";
                        cout<<"\n";
                        return 0;
                        }//end else
        }//end for
return uniqueref;
}

//Validation routine to check data has been entered
int PresVal(char presence[30])
{
//local variables
int length;

length=strlen(presence); //strlen extracts the number of characetrs entered and stores it under the variable lentgth
if(length==0)   //Checks to see if the length is 0 as if it is no data has been entered
        {
        //Error message is outputted if length = 0
        cout<<"Please enter a first name.";
        }//endif
return length;
}

//Validation routine to check data has been entered
int PresValL(char presence[30])
{
//local variable
int length;

length=strlen(presence); //strlen extracts the number of characetrs entered and stores it under the variable lentgth
if(length==0)     //Checks to see if the length is 0 as if it is no data has been entered
        {
        //Error message is outputted if length = 0
        cout<<"Please enter a last name";
        }//endif
return length;
}
```

```
//Validation routine to check data has been entered
int PresValO(char presence[30])
{
//local variable
int length;

length=strlen(presence);   //strlen extracts the number of characetrs entered and stores it under the variable lentgth
if(length==0)     //Checks to see if the length is 0 as if it is no data has been entered
        {
        //Error message is outputted if length = 0
        cout<<"Please enter address line 1";
        }//endif
return length;
}

//Validation routine to check data has been entered
int PresValT(char presence[30])
{
//local variable
int length;

length=strlen(presence);
if(length==0)
        {
        cout<<"Please enter adress line 2 ";
        }//endif
return length;
}

//Validation routine to check data has been entered
int PresValTh(char presence[30])
{
//local variable
int length;

length=strlen(presence); //strlen extracts the number of characetrs entered and stores it under the variable lentgth
if(length==0)   //Checks to see if the length is 0 as if it is no data has been entered
        {
        //Error message is outputted if length = 0
        cout<<"Please enter address line 3";
        }//endif
return length;
}

//Validation routine to ensure the postcode entered by the user conforms with the standard british formats for a postcode
int PostCodeVal(char postcode[9])
{
//local variables
int valid =0;
int length;

length=strlen(postcode);    //strlen extracts the number of characetrs entered and stores it under the variable lentgth

if(length==8)   //The postcode format: AA9A 9AA or AA99 9AA
```

```
{
if(isalpha(postcode[0]) && isalpha(postcode[1]) && isalpha(postcode[6]) && isalpha(postcode[7]))  //Checks certain positions are let
        {
        if(isdigit(postcode[2]) && isdigit(postcode[5]))  //Checks certain positions are numbers
                {
                if(isalpha(postcode[3])|| (isdigit(postcode[3])))
                        {
                        valid =1;          //Postcode format correct
                        }
                }//enddigcheck
                else
                        {
                        cout<<"\nPlease make sure all letters are in the correct place.";
                        }//end else - digit
        }//endlettererrorcheck
        else
                {
                cout<<"\nPlease make sure all digits are in the correct place.";
                }//end else - letter
}//endlengthcheck
else
        {
        if(length==7)   //The postcode format: A9A 9AA or A99 9AA or AA9 9AA
                {
                //A9A 9AA
                if(isalpha(postcode[0])&& isalpha(postcode[2]) && isalpha(postcode[5]) && isalpha(postcode[6]))  //Checks certain pos
                        {
                        if(isdigit(postcode[1]) && isdigit(postcode[4]))    //Checks certain positions are numbers
                                {
                                valid =1;
                                }//enddigitcheck
                                else  //A99 9AA
                                    {
                                    if(isdigit(postcode[2]))   //Checks certain position is a number
                                            {
                                            if(isalpha(postcode[0])&& isalpha(postcode[5]) && isalpha(postcode[6]))
                                                    {
                                                    if(isdigit(postcode[1]) && isdigit(postcode[4]))
                                                              {
                                                              valid =1;
                                                              }//end if
                                                            else
                                                                    {
                                                                    cout<<"\nPlease ensure digits are in the correct place.";
                                                                    }
                                                    }//end if - letter
                                                    else
                                                            {
                                                            cout<<"\nPlease ensure letters are in the correct place.";
                                                            }//end else
                                            }//endif for position 2 is number
                                            else
                                                {
                                                cout<<"\nPlease ensure letters are in the correct place.";
                                                }//end else
```

```
                        }//endletterelse
                }//endlettercheck
                  else   //AA9 9AA
                        {
                        if(isalpha(postcode[1]) || isdigit(postcode[2]))     //Checks certain position is a letter
                                {
                                if(isalpha(postcode[0])&& isalpha(postcode[5]) && isalpha(postcode[6]))
                                        {
                                        if(isdigit(postcode[2]) && isdigit(postcode[4]))
                                                {
                                                valid=1;
                                                }//end if - digit check
                                                else
                                                        {
                                                        cout<<"\nPlease ensure letters are in the correct place.";
                                                        }
                                        }//end if - letter check
                                        else
                                                {
                                                cout<<"\nPlease ensure digits are in the correct place.";
                                                }
                                }//end if for position 1 is letter
                                else
                                        {
                                        cout<<"\nPostcode invalid please try again.";
                                        }//end else
                        }//end else

        }//endlengthcheck
        else
        {
        if(length==6)    //Postcode format: A9 9AA
                {
                if(isalpha(postcode[0])&& isalpha(postcode[4]) && isalpha(postcode[5]))  //Checks certain positions are lett
                        {
                        if(isdigit(postcode[1]) && isdigit(postcode[3]))         //Checks certain positions are numbers
                                {
                                valid =1;
                                }
                                else
                                        {
                                        cout<<"\nPlease ensure letters are in the correct place.";
                                        }
                               }//endlettercheck
                               else
                                {
                                cout<<"\nPlease ensure digits are in the correct place.";
                                }
                }//endlength check
                else
                        {
                        cout<<"\n Please ensure the postcode is of a suitable length.";
                        }
                }//end if - length
}//endelse for length is 7
```

```
return valid;
}

//Validation routine to check that the mobile number entered is 11 digits long beginning '07'
int TelVal(char phonenum[12])
{

//local variables
int valid = 0;
int position;
int len=strlen(phonenum);

if(len==0)
        {
        valid=0;
        return valid;
        }

if(phonenum[0]=='0')     //Checks first digit is 0
        {
        if(phonenum[1]=='7')    //Checks second digit is 7
                {
                //Loops through the rest of the characters entered
                for(position=2;position<len;position++)
                        {
                        //Checks rest of the characters are numbers
                        if(isdigit(phonenum[position]))
                                {
                                valid = 1;
                                }//endif for position 10
                                else
                                        {
                                        //Error message if the number entered is not all digits
                                        cout<<"\nPlease ensure the mobile number is all digits.";
                                        valid=0;
                                        return valid;
                                        }//end else
                        }//endfor - checking each character is an integer
                }//endif - checking second digit is a 7
                else
                        {
                        //Error message is outputted to make sure mobile is in the correct format
                        cout<<"\nPlease make sure the phone number begins 07.";
                        }//end else
        }//end if - checking first digit is 0
        else
                {
                //Error message is outputted to make sure mobile is in the correct format
                cout<<"\nPlease enter a phone number beginning with 07";
                }//endelse

return valid;
}
```

```
//Validation routine to check that the mobile number entered is 11 digits long beginning '07'
int TelValE(char phonenum[12])
{
//local variables
int valid = 0;
int position;
int len=strlen(phonenum);

if(len==0)
        {
        valid=0;
        return valid;
        }

//Checks first digit is 0
if(phonenum[0]=='0')
        {
         //Checks second digit is 7
        if(phonenum[1]=='7')
                {
                //Loops through the rest of the characters entered
                for(position=2;position<len;position++)
                        {
                        //Checks rest of the characters are numbers
                        if(isdigit(phonenum[position]))
                                {
                                valid = 1;
                                }//endif for position 10
                                else
                                        {
                                        //Error message if the number entered is not all digits
                                        cout<<"\nPlease ensure the mobile number is all digits.";
                                        valid=0;
                                        return valid;
                                        }//end else
                        }//endfor - checking each character is an integer
                }//endif - checking second digit is a 7
                else
                        {
                        //Error message is outputted to make sure mobile is in the correct format
                        cout<<"\nPlease make sure the phone number begins 07.";
                        }//end else
        }//end if - checking first digit is 0
        else
                {
                //Error message is outputted to make sure mobile is in the correct format
                cout<<"\nPlease enter a phone number beginning with 07";
                }//endelse

return valid;
}


//Program which validates the ni number entered by the user
int ValidNI(char ninum[10])
```

```
{
int valid = 0;
int len = strlen(ninum);  //strlen extracts the number of characetrs entered and stores it under the variable len

if(len<13)
        {
        valid=0;
        return valid;
        }
        //Checks certain positions are spaces
        if(isspace(ninum[2]) && isspace(ninum[5]) && isspace(ninum[8]))
                {
                //Checks certain positions are letters
                if(isalpha(ninum[0]) && isalpha(ninum[1]) && isalpha(ninum[12]))
                        {
                         //Checks certain positions are numbers
                        if(isdigit(ninum[3]) && isdigit(ninum[4]) && isdigit(ninum[6]) && isdigit(ninum[7]) && isdigit(ninum[9]) && isdigit(n
                                {
                                valid = 1;
                                }// END DIGIT CHECK
                        else
                                {
                                //Error message is outputted if dgitis for a valid NI are in the wrong place
                                cout<<"\nDigits entered incorrectly, please ensure you have typed it in correct.";
                                }// end else for digit check error message
                        } // END LETTER CHECK
                else
                        {
                        //Error message is outputted if letters for a valid NI are in the wrong place
                        cout<<"\nLetters entered incorrectly, please ensure you have typed it in correct.";
                        }// end else for letter check error message
                } // END SPACE CHECK
        else
                {
                //Error message is outputted if NI is not in the correct format with spaces in the correct place
                cout<<"\nMake sure there are the correct spaces in the national insurance number";
                }// end else for space check error message

return valid;
}

//Validation routine to ensure the new username enetered is unique and not used by another member of staff
int UniqueUser(char user[15])
{
//local variables
int userok=1;
int find;
int compare;
int len;
len = strlen(user); //strlen extracts the number of characetrs entered and stores it under the variable len


ReadBackStaffFile();

//Checks to see if the length is 0 as if it is no data has been entered
```

```
if(len==0)
        {
        userok=0;
        //If the length of the input is 0, error message is outputted
        cout<<"\n Please ensure a username was entered.";
        }

for (find = 0; find < nsi; find++)    //searches through the staff file
        {
        compare = strcmpi(username[find], user);   //Compares all usernames in the staff file with username enetered by the user
        if (compare == 0)    //If compare==0 that means there is a match so username is already in use
                {
                //If userok is set equal to 0 username has been found already in the file
                userok = 0;
                //Error message is outputted to tell the user this usernme cannot be used
                cout<<"\n Username already in use. Please try again.";
                } //endif
        }//endfor
return userok;
}

//Validation routine which checks that the password enetered meets certain rules
int PasswordVal(char pass[10])
{
//local variables
int valid = 0;
int length = strlen(pass);  //strlen extracts the number of characetrs entered and stores it under the variable length
int checkpunc;

if(length >=8) //Checks that the password if of length 8 or greater
        {
        //Loops through all characters in the password
        for(checkpunc=0;checkpunc<length;checkpunc++)
                {
                //Loops the length of the password to check that at least one character is a piece of punctuation
                if(ispunct(pass[checkpunc]))
                        {
                        valid = 1;
                        }  // end if - punctuation check

                }//end for

        }// end if - length check
if(valid!=1)
        {
        //If one of the requirements in not met, error message is outputted
        cout<<"\nPassword not valid. Please ensure it is minimum 8 characters with at least one piece of punctuation.";
        }

return valid;
}

//Validation routine which checks that the level of access enetered is correct within a certain range
int RangeCheck(char level[3])
{
```

```
//local variables
int valid = 0;
int intlev;
int len = strlen(level);
intlev = atoi(level);       //converts level entered into an integer for range check
int position;

if(len==0)
        {
        cout<<"\nPlease ensure an access level was entered.";
        valid=0;
        return valid;
        }//end if

for(position=0;position<len;position++)
        {
        if(isdigit(level[position]))
                {
                if(intlev <=3 && intlev>=1)       // checks that the level of access entered is between 1 and 3
                        {
                        valid = 1;
                        } // end if
                else
                        {
                        //If level of access is not within this range error message is outputted
                        cout<<"\n Please choose a level of access between 1 and 3.";
                        }//end else
                }//end if - digit check
                else
                        {
                        cout<<"\n Please ensure you have entered a number 1-3.";
                        }//end else
        }//end for
return valid;
}
//This function determines whether or not the staff menu should be outputted to the screen
int ChangeStaff()
{
int changestatus=0;
//While loop to output the Change Staff Menu to the screen
while(changestatus != 6)
        {
        changestatus = ChangeStaffMenu();
        }//endwhile
getch();
return 0;
}
//Outputs change staff menu options to the user and allows them to choose one of them
int ChangeStaffMenu()
{
//Routine that allows a user to enter different functions for changing information about a staff member
int changestaffchoice;
//Outputs menu to the screen
cout<<"\n \t Change Staff Information";
cout<<"\n \t ***********************\n \n";
```

```
cout<<"\n \t 1. Change Home Address";
cout<<"\n \t 2. Change Mobile Number";
cout<<"\n \t 3. Change Username";
cout<<"\n \t 4. Change Password";
cout<<"\n \t 5. Change Level of Access";
cout<<"\n \t 6. Return to Staff Menu";
cout<<"\n \t Enter choice: ";
getch();
cin>>changestaffchoice;  //Input from the user for what they would like to access
clrscr();
switch(changestaffchoice) //Allows different functions to be run depending on the menu option entered by the user
        {
        case 1:  //If changestaffchoice =1 then the SHomeAdd function is ran
             {
             SHomeAdd();
             break;
             }
        case 2: //If changestaffchoice =2 then the STelNum function is ran
             {
             STelNum();
             break;
             }
        case 3:   //If changestaffchoice =3 then the ChgUsername function is ran
             {
             if(level==3)     //Only allows access to the fuunction if the user logged on has level of access 3
                {
                ChgUsername();
                }//end if
             else
                {
                //If level of access is lower, then the following error message is outputted
                cout<<"\n Not authorised.";
                }//end else
             break;
             }
        case 4:  //If changestaffchoice =4 then the ChgPassword function is ran
             {
             if(level==3) //Only allows access to the fuunction if the user logged on has level of access 3
                {
                ChgPassword();
                }//end if
             else
                {
                //If level of access is lower, then the following error message is outputted
                cout<<"\n Not authorised.";
                }//end else
             break;
             }
        case 5: //If changestaffchoice =5 then the ChgLoa function is ran
             {
             if(level==3) //Only allows access to the fuunction if the user logged on has level of access 3
                {
                ChgLoa();
                }//end if
             else
```

```
                {
                //If level of access is lower, then the following error message is outputted
                cout<<"\n Not authorised";
                }//end else
            break;
            }
        case 6: //If changestaffchoice =6 then this is quit so returns to staff menu
            {
            break;
            }
        default:
              {
              //If an incorrect number is entered the following error message is ouputted
              cout<<"\nPlease enter a number between 1 and 6.";
              getch();
              break;
              }
        }//endcase

clrscr();  //Clears menu from the screen
getch();
return changestaffchoice;
}

//Allows the user to change the home address associated with a staff member
int SHomeAdd()
{
//local variables
char looking[25];
int compare;
int find;
char result[2];
int compresult;

int adone=0;
int adthree=0;
int pcode=0;

ReadBackStaffFile();
cin.get();

cout<<"\nChange Home Address";
cout<<"\n********************";

cout<<"\nEnter the staff reference of the staff member's home address you wish to change: ";
cin.getline(looking,25);  //Staff member to be searched for
for(find =0; find<nsi; find++)  //Searches through all the staff in the file
        {
        compare = strcmpi(looking, staffref[find]);    //Comapres staff reference in the file with the staff reference entered
        if(compare ==0)  //If compare ==0 then there is a staff reference in the file the same as the one entered by the user
                {
                //If staff reference matches, details are outputted to the user to confirm it is the correct member of staff
                cout<<"\nFull name: "<<fnamestaff[find]<<" "<<lnamestaff[find];
                cout<<"\nMobile Number: "<<telnostaff[find];
                cin.get();
```

```
                cout<<"\nIs this the correct member of staff to be amended (Y/N): ";
                cin>>result;   //User confirms whether or not it is the correct staff member
                compresult=strcmpi(result,"Y"); //Compares the input with Y
                if(compresult == 0) //If the input is Y then the new address can be added, If input is N returns to Change Staff Menu
                        {
                        //New address is entered and is validated upon entry
                        while(adone == 0)    //while loop that controls validation to ensure address line 1 is entered
                                {
                                cin.get();
                                cout<<"\nEnter address line 1: ";
                                cin.getline(oneadstaff[find],30);
                                adone = PresValO(oneadstaff[find]);
                                }//end while

                                cout<<"\nEnter address line 2: ";   //No validation as address line 2 is not always included in an address
                                cin.getline(twoadstaff[find],30);


                        while(adthree == 0)  //while loop that controls validation to ensure address line 3 is entered
                                {
                                cout<<"\nEnter address line 3: ";
                                cin.getline(threeadstaff[find],30);
                                adthree = PresValTh(threeadstaff[find]);
                                }// end while

                        while(pcode==0)   //while loop that controls validation to ensure the postcode entered is a valid UK postcode
                                {
                                cout<<"\nEnter postcode: ";
                                cin.getline(pcodestaff[find],9);
                                cout<<"\n";
                                pcode = PostCodeVal(pcodestaff[find]);
                                }//endwhile
                        ReWriteStaffFile();
                        return 0;                                   //New details are saved to the file
                        }// end if
                        return 0;
                }//end if
        }//end for
if(compare!=0)
        {
        //If no match is found between the refernce entered and those in the file error messsage is outputted
        cout<<"\nStaff reference not found.";    //If staff reference entered is not found in the file - this message is outputted
        }//end if
getch();
return 0;
}

//Allows the user to change the telephone number associated with a staff member
int STelNum()
{
//local variables
char looking[25];
int compare;
int find;
char result[2];
```

```
int compresult;
int tel=0;

ReadBackStaffFile();
cin.get();
cout<<"\nChange Mobile Number";
cout<<"\n**********************";

cout<<"\nEnter the staff reference of the staff member's mobile number you wish to change: ";
cin.getline(looking,25);  //Staff member to be searched for
for(find =0; find<nsi; find++)     //Searches through all the staff in the file
        {
        compare = strcmpi(looking, staffref[find]);    //Compares the staff reference entered with the staff references in the file
        if(compare ==0)  //If compare ==0 then there is a staff reference in the file the same as the one entered by the user
                {
                 //If a match is found, details are outputted to check that it is the correct user
                cout<<"\nFull name: "<<fnamestaff[find]<<" "<<lnamestaff[find];
                cout<<"\nMobile Number: "<<telnostaff[find];
                cin.get();
                cout<<"\nIs this the correct member of staff to be amended (Y/N): ";
                cin>>result;     //User confirms whether or not it is the correct staff member
                compresult=strcmpi(result,"Y");   //Compares the input with Y
                if(compresult == 0)  //If the input is Y then the new mobile number can be added, If input is N returns to Change Staff Menu
                        {
                        cin.get();
                        //New number is entered and is validated upon entry
                        while(tel==0)  //while loop that controls validation to ensure the mobile number entered is of the correct format: 0
                          {
                          cout<<"\nEnter mobile number: ";
                          cin.getline(telnostaff[find], 12);
                          cout<<"\n";
                          tel = TelVal(telnostaff[find]);
                          }//end while
                        ReWriteStaffFile();     //New details are saved to the file
                        return 0;
                        }// end if
                        return 0;
                }//end if
           }//end for
if(compare!=0)
        {
        //If no match is found between the refernce entered and those in the file error messsage is outputted
        cout<<"\nStaff reference not found.";
        }//end if
getch();
return 0;
}

//Routine that updates the level of access for a user
int ChgLoa()
{
//local variables
char looking[3];
int compare;
int find;
```

```
char result[2];
int compresult;
int levrange=0;

ReadBackStaffFile();
cin.get();

cout<<"\nChange level of access";
cout<<"\n**********************";

cout<<"\nEnter the staff reference of the staff member's access level you wish to change: ";
cin.getline(looking,3);         //Staff member to be searched for
for(find=0; find<nsi; find++)                  //Searches through all the staff in the file
        {
        compare = strcmpi(looking, staffref[find]); //Compares the staff reference entered with the staff references in the file
        if(compare==0)   //If compare ==0 then there is a staff reference in the file the same as the one entered by the user
                {
                //If a match is found, details are outputted to check that it is the correct user
                cout<<"\nFull name: "<<fnamestaff[find]<<" "<<lnamestaff[find];
                cout<<"\nCurrent level of access: "<<loa[find];
                cin.get();
                cout<<"\nIs this the correct member of staff to be amended (Y/N): ";
                cin>>result; //User confirms whether or not it is the correct staff member
                compresult=strcmpi(result,"Y");  //Compares the input with Y
                if(compresult == 0)  //If the input is Y then the new access level can be added, If input is N returns to Change Staff Menu
                        {
                        cin.get();
                        //Adds new level of access and it is validated upon entry
                        while(levrange == 0)  //while loop that controls validation to ensure the level of access entered is between 1 and 3
                                {
                                cout<<"\nEnter level of access: ";
                                cin.getline(loa[find],2);
                                levrange = RangeCheck(loa[find]);
                                }// end while
                        ReWriteStaffFile();    //New details are saved to the file
                        return 0;
                        }// end if
                        return 0;
                }//end if
           }//end for
if(compare!=0)
        {
        //If no match is found between the refernce entered and those in the file error messsage is outputted
        cout<<"\nStaff reference not found.";
        }//end if
getch();
return 0;
}

//Allow a user to change the username of a staff member stored in the file
int ChgUsername()
{
//local variables
char looking[25];
int compare;
```

```
int compresult;
int find;
char result[2];
int uquser=0;
char newusername[15];

ReadBackStaffFile();
cin.get();

cout<<"\nChange Username";
cout<<"\n***************";
cout<<"\nEnter the staff reference of the staff member's username you wish to change: ";
cin.getline(looking,25);     //Staff member to be searched for
for(find =0; find<nsi; find++)      //Searches through all the staff in the file
        {
        compare = strcmpi(looking, staffref[find]); //Compares the staff reference entered with the staff references in the file
        if(compare ==0)  //If compare ==0 then there is a staff reference in the file the same as the one entered by the user
                {
                //If a match is found, details are outputted to check that it is the correct user
                cout<<"\nFull name: "<<fnamestaff[find]<<" "<<lnamestaff[find];
                cout<<"\nCurrent username: "<<username[find];
                cin.get();
                cout<<"\nIs this the correct member of staff to be amended (Y/N): ";
                cin>>result; //User confirms whether or not it is the correct staff member
                compresult=strcmpi(result,"Y"); //Compares the input with Y
                if(compresult==0) //If the input is Y then the new username can be added, If input is N returns to Change Staff Menu
                        {
                        cin.get();
                        //Allows new username to be added - it is valideated upon entry
                        while(uquser == 0)    //while loop that controls validation to ensure the username is unique
                                {
                                cout<<"\nEnter new username: ";
                                cin.getline(newusername,15);
                                uquser = UniqueUser(newusername);
                                strcpy(username[find],newusername);
                                }// end while
                        ReWriteStaffFile();   //New details are saved to the file
                        return 0;
                        }// end if
                        return 0;
                }//end if
           }//end for
if(compare!=0)
        {
        //If no match is found between the refernce entered and those in the file error messsage is outputted
        cout<<"\nStaff reference not found.";
        }//end if
getch();
return 0;
}

//Routine that allows a user to change a member of staffs password
int ChgPassword()
{
//local variables
```

```
char looking[25];
int compare;
int compresult;
int find;
char result[2];
int passval=0;

ReadBackStaffFile();
cin.get();

cout<<"\nChange Password";
cout<<"\n***************";
cout<<"\nEnter the staff reference of the staff member's password you wish to change: ";
cin.getline(looking,25);   //Staff member to be searched for
for(find =0; find<nsi; find++)    //Searches through all the staff in the file
        {
        compare = strcmpi(looking, staffref[find]);   //Compares the staff reference entered with the staff references in the file
        if(compare ==0)   //If compare ==0 then there is a staff reference in the file the same as the one entered by the user
                {
                //If a match is found, details are outputted to check that it is the correct user
                cout<<"\nFull name: "<<fnamestaff[find]<<" "<<lnamestaff[find];
                cout<<"\nMobile Number: "<<telnostaff[find];
                cin.get();
                cout<<"\nIs this the correct member of staff to be amended (Y/N): ";
                cin>>result;  //User confirms whether or not it is the correct staff member
                compresult=strcmpi(result,"Y"); //Compares the input with Y
                //If the input is Y then the new username can be added, If input is N returns to Change Staff Menu
                if(compresult == 0)
                        {
                        cin.get();
                        //Adds the new password and validates it upon entry
                        while(passval==0) //while loop that controls validation to ensure the password conforms to a set of rules
                                {
                                cout<<"\nEnter new password (It must be minimum 8 characters long with at least one piece of punctuation): "
                                cin.getline(password[find],15);
                                passval = PasswordVal(password[find]);
                                }//end while
                        ReWriteStaffFile();  //New details are saved to the file
                        return 0;
                        }// end if
                        return 0;
                }//end if
           }//end for
if(compare!=0)
        {
        //If no match is found between the refernce entered and those in the file error messsage is outputted
        cout<<"\nStaff reference not found.";
        }//end if
getch();
return 0;
}


//Deletes a staff member from the staff file
int DeleteStaff()
```

```
{
//local variables
int find;
int compare;
int compresult;
int del;
char result[2];
char looking[25];

ReadBackStaffFile();
cin.get();

cout<<"\nDelete Staff";
cout<<"\n************";
cout<<"\n";
cout<<"\nEnter the staff reference of the staff member you wish to delete: ";
cin.getline(looking,25);  //Staff reference to be searched for
for(find=0;find<nsi;find++)  //Searches through all the staff in the file
        {
        compare = strcmpi(looking,staffref[find]);  //Coampres each staff reference in the file with the one entered by the user
        if(compare==0)   //If compare ==0 then there is a staff reference in the file the same as the one entered by the user
                {
                //If a match is found, details about that member of staff are outputted to ensure it is the correct person
                cout<<"\nFull name: "<<fnamestaff[find]<<" "<<lnamestaff[find];
                cout<<"\nMobile Number: "<<telnostaff[find];
                cin.get();
                cout<<"\nIs this the correct member of staff to be deleted (Y/N): ";
                cin>>result; //User confirms whether or not it is the correct staff member
                compresult=strcmpi(result,"Y");  //Compares the input with Y
                if(compresult == 0) //If the input is Y then the new username can be added, If input is N returns to Change Staff Menu
                        {
                        //Loops through and deletes all informarion for that staff member
                        for(del=find;del<nsi;del++)
                                {
                                strcpy(staffref[del], staffref[del+1]);
                                strcpy(fnamestaff[del], fnamestaff[del+1]);
                                strcpy(lnamestaff[del], lnamestaff[del+1]);
                                strcpy(oneadstaff[del], oneadstaff[del+1]);
                                strcpy(twoadstaff[del], twoadstaff[del+1]);
                                strcpy(threeadstaff[del], threeadstaff[del+1]);
                                strcpy(pcodestaff[del], pcodestaff[del+1]);
                                strcpy(telnostaff[del], telnostaff[del+1]);
                                strcpy(ninum[del], ninum[del+1]);
                                strcpy(username[del], username[del+1]);
                                strcpy(password[del], password[del+1]);
                                strcpy(loa[del], loa[del+1]);
                                nsi=nsi -1;       //Decreases the number of staff stored in the file by one
                                itoa(nsi,nsc,10);
                                ReWriteStaffFile();  //Updates file
                                }// end for
                        }//end if
                        return 0;
                }//end if
        }//end for
if(compare!=0)
```

```
        {
        //If staff reference entered is not found in the file - this error message is outputted
        cout<<"\nStaff reference not found.";
        }//end if

getch();
return 0;
}

//Allows a user to view information about a staff member by entering the staff reference
int ViewStaff()
{
//Local variables
char looking[25];
int compare;
int find;

cout<<"\nView Staff Member by Staff Reference";
cout<<"\n***********************************";
ReadBackStaffFile();
cin.get();

cout<<"\nEnter staff reference: ";
cin.getline(looking,25);      //Staff reference to be searched
for(find=0;find<nsi;find++)     //Searches through all the staff in the file
        {
        compare = strcmpi(looking,staffref[find]);    //Compares staff reference entered by the user with staff references in the file
        if(compare == 0)  //If compare ==0 there is match between the reference entered and the one stored in the file
                {
                //THE staff information stored under this reference is then outputted to the screeen
                cout<<"\nName: "<<fnamestaff[find]<<" "<<lnamestaff[find];
                cout<<"\n";
                cout<<"\nAddress: "<<oneadstaff[find];
                cout<<"\n         "<<twoadstaff[find];
                cout<<"\n         "<<threeadstaff[find];
                cout<<"\n         "<<pcodestaff[find];
                cout<<"\n";
                cout<<"\Mobile number: "<<telnostaff[find];
                cout<<"\n";
                cout<<"\nEmergency contact number: "<<emtel[find];
                cout<<"\n";
                cout<<"\nNational Insurance Number: "<<ninum[find];
                }// end if
        }// end for
if(compare!=0)
        {
        //If staff reference entered is not found in the file - this error message is outputted
        cout<<"\nStaff reference not found.";
        }//end if
getch();
return 0;
}

//Saves a copy of all files to a USB
int Backup()
```

```
{
cout<<"\n\tBacking up files.\n"; //backup to usb memory drive

system("copy StaffFileSD D:\\ProgramFileBackup\\");

system("copy CustomerFileSD D:\\ProgramFileBackup\\");

system("copy QuotesFileSD D:\\ProgramFileBackup\\");

system("copy InvoiceFileSD D:\\ProgramFileBackup\\");

system("copy StockFileSD D:\\ProgramFileBackup\\");

system("copy ScheduleFileSD D:\\ProgramFileBackup\\");

cout<<"\n\tBack up attempt complete, if the backup has failed\n\tplease check the USB storage drive has the folder\n\t";
getch();
return 0;
}

//Recovers all lost files from the USB
int Recovery()
{
char confirm[2];

int compare;

cout<<"\n Recovery of data files.\n"; //recovery from USB
cout<<"\n ************************";

cout<<"\n Enter Y to recover the file N to not recover that file. \n";

cin.get();
cout<<"\n Staff File: Comfirm recovery: ";
cin.getline(confirm,2);

compare=strcmpi(confirm,"Y");

if (compare==0)

        {
        system("copy StaffFileSD C:\\ProgramFileBackup\\");
        }//endif - recovery for staff file

cout<<"\n Customer File: Comfirm recovery: ";
cin.getline(confirm,2);

compare=strcmpi(confirm,"Y");

if (compare==0)

        {
        system("copy CustomerFileSD C:\\ProgramFileBackup\\");
        }//endif - recovery for customer file
```

```
cout<<"\n Quote File: Comfirm recovery: ";
cin.getline(confirm,2);

compare=strcmpi(confirm,"Y");

if (compare==0)
        {
        system("copy QuotesFileSD C:\\ProgramFileBackup\\");
        }//endif - recovery for quotes file

cout<<"\n Invoice File: Comfirm recovery: ";
cin.getline(confirm,2);

compare=strcmpi(confirm,"Y");

if (compare==0)

        {
        system("copy InvoiceFileSD C:\\ProgramFileBackup\\");
        }//endif - recovery for invoice file

cout<<"\n Stock File: Comfirm recovery: ";
cin.getline(confirm,2);

compare=strcmpi(confirm,"Y");

if (compare==0)

        {
        system("copy StockFileSD C:\\ProgramFileBackup\\");
        }//endif - recovery for stock file

cout<<"\n Schedule File: Comfirm recovery: ";
cin.getline(confirm,2);

compare=strcmpi(confirm,"Y");

if (compare==0)
        {
        system("copy ScheduleFileSD C:\\ProgramFileBackup\\");
        }// endif - recovery for schedule file


getch();
return 0;
}



//Reads all staff information from the file
int ReadBackStaffFile()
{
int count;
ifstream fin(FileName1,ios::binary);
fin.read((char*)&nsc, sizeof(nsc));
```

```
sscanf(&nsc[0], "%d", &nsi);
for(count=0;count<nsi;count++)
        {
        fin.getline(staffref[count],sizeof(staffref[count]));
        fin.getline(fnamestaff[count],sizeof(fnamestaff[count]));
        fin.getline(lnamestaff[count],sizeof(lnamestaff[count]));
        fin.getline(oneadstaff[count],sizeof(oneadstaff[count]));
        fin.getline(twoadstaff[count],sizeof(twoadstaff[count]));
        fin.getline(threeadstaff[count],sizeof(threeadstaff[count]));
        fin.getline(pcodestaff[count],sizeof(pcodestaff[count]));
        fin.getline(telnostaff[count],sizeof(telnostaff[count]));
        fin.getline(emtel[count], sizeof(emtel[count]));
        fin.getline(ninum[count],sizeof(ninum[count]));
        fin.getline(username[count],sizeof(username[count]));
        fin.getline(password[count],sizeof(password[count]));
        fin.getline(loa[count],sizeof(loa[count]));
        }//end for
fin.close();
return 0;
}

//Writes in new staff information to the file
int ReWriteStaffFile()
{
int count;
ofstream fout(FileName1, ios::binary);
fout.write((char*)&nsc,sizeof(nsc));
for(count=0;count<nsi;count++)
        {
        fout.write((char*)&staffref[count],strlen(staffref[count]));
        fout.write("\n",1);
        fout.write((char*)&fnamestaff[count],strlen(fnamestaff[count]));
        fout.write("\n",1);
        fout.write((char*)&lnamestaff[count],strlen(lnamestaff[count]));
        fout.write("\n",1);
        fout.write((char*)&oneadstaff[count],strlen(oneadstaff[count]));
        fout.write("\n",1);
        fout.write((char*)&twoadstaff[count],strlen(twoadstaff[count]));
        fout.write("\n",1);
        fout.write((char*)&threeadstaff[count],strlen(threeadstaff[count]));
        fout.write("\n",1);
        fout.write((char*)&pcodestaff[count],strlen(pcodestaff[count]));
        fout.write("\n",1);
        fout.write((char*)&telnostaff[count],strlen(telnostaff[count]));
        fout.write("\n",1);
        fout.write((char*)&emtel[count], strlen(emtel[count]));
        fout.write("\n",1);
        fout.write((char*)&ninum[count],strlen(ninum[count]));
        fout.write("\n",1);
        fout.write((char*)&username[count],strlen(username[count]));
        fout.write("\n",1);
        fout.write((char*)&password[count],strlen(password[count]));
        fout.write("\n",1);
        fout.write((char*)&loa[count],strlen(loa[count]));
        fout.write("\n",1);
```

```
        }//end for
fout.close();
return 0;
}

//This function determines whether or not the customer menu should be outputted to the screen
int Customers()
{
int customerstatus=0;
//While loop to output the customer menu
while(customerstatus != 5)
        {
        customerstatus = CustomerMenu();
        }//endwhile
getch();
return 0;
}

//Outputs the customer menu to the screen to allow a user to chose an option
int CustomerMenu()
{
int customerchoice;
//Menu is outputtef to the screen
cout<<"\n \t Customer Menu";
cout<<"\n \t *************\n \n";
cout<<"\n \t 1. Add Customer";
cout<<"\n \t 2. Change customer information";
cout<<"\n \t 3. Delete customer";
cout<<"\n \t 4. View customer";
cout<<"\n \t 5. Return to Main Menu";
cout<<"\n";
cout<<"\n \t Enter choice: ";
getch();
cin>>customerchoice;   //Inputted menu choice from user
clrscr();
switch(customerchoice)  //Allows different menu options to be ran depending on the menu option entered by the user
        {
        case 1: //If customerchoice =1 then the AddCustomer function is ran
             {
             AddCustomer();
             break;
             }
        case 2: //If customerchoice =2 then the ChangeCustomer function is ran
             {
             ChangeCustomer();
             break;
             }

        case 3:  //If customerchoice =3 then the DeleteCustomer function is ran
             {
             if(level>=2)     //Only allows access to the fuunction if the user logged on has level of access 2 or 3
                {
                DeleteCustomer();
                }
             else
```

```
                {
                //If level of access is lower, then the following error message is outputted
                cout<<"\nNot authorised.";   //If they don't - message will be outputted
                getch();
                }
           break;
           }

      case 4: //If customerchoice =4 then the ViewCustomer function is ran
           {
           ViewCustomer();
           break;
           }

      case 5:   //If customerchoice =5 then returns to the main menu
           {
           break;
           }
      default:
             {
             //If users input is not a menu option 1-6 then the following error message is outputted
             cout<<"\nPlease enter a number between 1 and 5.";
             getch();
             break;
             }
      }//endcase


clrscr();
getch();
return customerchoice;
}

//Allows a user to add a customer to the customer file
int AddCustomer()
{
//Local variables

int tpres=0;
int fname=0;
int lname=0;
int adone=0;
int adthree=0;
int post=0;
int tel=0;
int ref=0;
char tempcust[3];

char telb[30];
int telbuff=0;

cout<<"\nAdd Customer";
cout<<"\n************";
while(ref==0)
      {
```

```
cin.get();
cout<<"\nEnter new customer reference: ";
cin.getline(tempcust,3);            //customer reference to be added
ref = CustRefVal(tempcust);
if(ref!=0)
        {
        ofstream fout(FileName2,ios::in);
        fout.seekp(custref*sizeof(a_cust));
        cin.get();
        //All validation routines for information regarding a customer being entered
        while(tpres==0)                  //while loop that controls validation to ensure a title is entered
                {
                cout<<"\nEnter title: ";
                cin.getline(a_cust.title,10);
                tpres = PresValTitle(a_cust.title);
                } //end while
        while(fname==0)       //while loop that controls validation to ensure the first name is entered
                {
                cout<<"\nEnter first name: ";
                cin.getline(a_cust.fnamecust,15);
                fname = PresValFirst(a_cust.fnamecust);
                }  // end while
        while(lname ==0)              //while loop that controls validation to ensure the last name is entered
                {
                cout<<"\nEnter last name: ";
                cin.getline(a_cust.lnamecust,15);
                lname = PresValLast(a_cust.lnamecust);
                }// end while
        while(adone==0)          //while loop that controls validation to ensure address line 1 is entered
                {
                cout<<"\nEnter address line 1: ";
                cin.getline(a_cust.oneadcust,30);
                adone = PresValOne(a_cust.oneadcust);
                }// end while

cout<<"\nEnter address line 2: ";   //No validation on address line 2 as its not mandatory for an address
cin.getline(a_cust.twoadcust,30);

        while(adthree==0)        //while loop that controls validation to ensure address line 3 is entered
                {
                cout<<"\nEnter address line 3: ";
                cin.getline(a_cust.threeadcust,15);
                adthree = PresValThree(a_cust.threeadcust);
                }// end while
        while(post==0)  //while loop that controls validation to ensure the postcode endered is a valid UK postcode
                {
                cout<<"\nEnter postcode: ";
                cin.getline(a_cust.pcodecust,9);
                post = PostcodeValCust(a_cust.pcodecust);
                }// end while
        while(telbuff==0 || tel==0)  //while loop that controls validation to ensure the mobile number is in the correct formst: 07N
                {
                cout<<"\nEnter mobile number: ";
                cin.getline(telb,30);
                telbuff = Buffer(telb,11);
```

```
                        if(telbuff!=0 && tel==0)
                                {
                                tel = TelValCust(telb);
                                }
                        }// end while
                strcpy(a_cust.telnocust,telb);
                strcpy(a_cust.flag,"1");        //Once data is added, changes the value of the flag to 1 to show that data is now stored at
                fout.write((char*)&a_cust,sizeof(a_cust));
                fout.close();
                return 0;
                }//endif
        }//endwhile


getch();
return 0;
}

//Validation routine to check a sensible reference has been entered
int CustRefVal(char cust[3])
{
//local variables
int length;
int valid=0;
int position;
int compare;


custref = atoi(cust);
ifstream fin(FileName2,ios::binary);
fin.seekg(custref*sizeof(a_cust));
fin.get((char*)&a_cust,sizeof(a_cust));
fin.close();
compare = strcmpi(a_cust.flag, "0");       //Checks that there is no data already stored at this location in the file
if(compare!=0)  //Means there is already data stored at this location
                        {
                        //Error message is outputted
                        cout<<"\n Please enter a different customer reference.";
                        valid=0;
                        }//end if

length = strlen(cust);

if(length==0)
        {
        cout<<"\nPlease ensure a reference has been entered.";
        valid=0;
        }

if(compare==0)
        {
        for(position=0;position<length;position++)
                {
                if(isdigit(cust[position]))
                        {
```

```
                        valid=1;
                        }//end if - digit check
                else
                        {
                        cout<<"\n Please ensure you have entered a number for the reference.";
                        valid=0;
                        }//end else
                }//end for
        }//end if

getch();
return valid;
}

//Validation routine to check data has been entered
int PresValTitle(char presence[30])
{
int length;
length=strlen(presence);    //strlen extracts the number of characetrs entered and stores it under the variable length
if(length==0)  //Checks to see if the length is 0 as if it is no data has been entered
        {
        //Error message is outputted if length = 0
        cout<<"Please enter a title.";
        }//endif
return length;
}

//Validation routine to check data has been entered
int PresValFirst(char presence[30])
{
int length;
length=strlen(presence); //strlen extracts the number of characetrs entered and stores it under the variable length
if(length==0)   //Checks to see if the length is 0 as if it is no data has been entered
        {
        //Error message is outputted if length = 0
        cout<<"Please enter a first name.";
        }//endif
return length;
}

//Validation routine to check data has been entered
int PresValLast(char presence[30])
{
int length;
length=strlen(presence);  //strlen extracts the number of characetrs entered and stores it under the variable length
if(length==0)   //Checks to see if the length is 0 as if it is no data has been entered
        {
        //Error message is outputted if length = 0
        cout<<"Please enter a last name.";
        }//endif
return length;
}

//Validation routine to check data has been entered
int PresValOne(char presence[30])
```

```
{
int length;
length=strlen(presence);  //strlen extracts the number of characetrs entered and stores it under the variable length
if(length==0)   //Checks to see if the length is 0 as if it is no data has been entered
        {
        //Error message is outputted if length = 0
        cout<<"Please enter address line 1.";
        }//endif
return length;
}

//Validation routine to check data has been entered
int PresValTwo(char presence[30])
{
int length;
length=strlen(presence); //strlen extracts the number of characetrs entered and stores it under the variable length
if(length==0)  //Checks to see if the length is 0 as if it is no data has been entered
        {
        //Error message is outputted if length = 0
        cout<<"Please enter data";
        }//endif
return length;
}

//Validation routine to check data has been entered
int PresValThree(char presence[30])
{
int length;
length=strlen(presence); //strlen extracts the number of characetrs entered and stores it under the variable length
if(length==0) //Checks to see if the length is 0 as if it is no data has been entered
        {
        //Error message is outputted if length = 0
        cout<<"Please enter address line 3.";
        }//endif
return length;
}

//Validation routine to ensure the postcode entered by the user conforms with the standard british formats for a postcode
int PostcodeValCust(char postcode[9])
{
//local variables
int valid =0;
int length;

length=strlen(postcode);    //strlen extracts the number of characetrs entered and stores it under the variable lentgth

if(length==8)   //The postcode format: AA9A 9AA or AA99 9AA
        {
        if(isalpha(postcode[0]) && isalpha(postcode[1]) && isalpha(postcode[6]) && isalpha(postcode[7]))  //Checks certain positions are let
                {
                if(isdigit(postcode[2]) && isdigit(postcode[5]))  //Checks certain positions are numbers
                        {
                        if(isalpha(postcode[3])|| (isdigit(postcode[3])))
                                {
                                valid =1;          //Postcode format correct
```

```
                        }
                }//enddigcheck
                else
                        {
                        cout<<"\nPlease make sure all letters are in the correct place.";
                        }//end else - digit
        }//endlettererrorcheck
        else
                {
                cout<<"\nPlease make sure all digits are in the correct place.";
                }//end else - letter
}//endlengthcheck
else
        {
        if(length==7)   //The postcode format: A9A 9AA or A99 9AA or AA9 9AA
                {
                //A9A 9AA
                if(isalpha(postcode[0])&& isalpha(postcode[2]) && isalpha(postcode[5]) && isalpha(postcode[6]))  //Checks certain pos
                        {
                        if(isdigit(postcode[1]) && isdigit(postcode[4]))    //Checks certain positions are numbers
                                {
                                valid =1;
                                }//enddigitcheck
                                else  //A99 9AA
                                    {
                                    if(isdigit(postcode[2]))   //Checks certain position is a number
                                            {
                                            if(isalpha(postcode[0])&& isalpha(postcode[5]) && isalpha(postcode[6]))
                                                    {
                                                    if(isdigit(postcode[1]) && isdigit(postcode[4]))
                                                              {
                                                              valid =1;
                                                              }//end if
                                                            else
                                                                    {
                                                                    cout<<"\nPlease ensure digits are in the correct place.";
                                                                    }
                                                    }//end if - letter
                                                    else
                                                            {
                                                            cout<<"\nPlease ensure letters are in the correct place.";
                                                            }//end else
                                            }//endif for position 2 is number
                                            else
                                                {
                                                cout<<"\nPlease ensure letters are in the correct place.";
                                                }//end else
                                    }//endletterelse
                        }//endlettercheck
                          else   //AA9 9AA
                                {
                                if(isalpha(postcode[1]) || isdigit(postcode[2]))     //Checks certain position is a letter
                                        {
                                            if(isalpha(postcode[0])&& isalpha(postcode[5]) && isalpha(postcode[6]))
                                                    {
```

```
                                        if(isdigit(postcode[2]) && isdigit(postcode[4]))
                                                {
                                                valid=1;
                                                }//end if - digit check
                                                else
                                                        {
                                                        cout<<"\nPlease ensure letters are in the correct place.";
                                                        }
                                        }//end if - letter check
                                        else
                                                {
                                                cout<<"\nPlease ensure digits are in the correct place.";
                                                }
                                }//end if for position 1 is letter
                                else
                                        {
                                        cout<<"\nPostcode invalid please try again.";
                                        }//end else
                        }//endlengthcheck
                        else
                        {
                        if(length==6)    //Postcode format: A9 9AA
                                {
                                if(isalpha(postcode[0])&& isalpha(postcode[4]) && isalpha(postcode[5]))  //Checks certain positions are lett
                                        {
                                        if(isdigit(postcode[1]) && isdigit(postcode[3]))         //Checks certain positions are numbers
                                                {
                                                valid =1;
                                                }
                                                else
                                                        {
                                                        cout<<"\nPlease ensure letters are in the correct place.";
                                                        }
                                               }//endlettercheck
                                               else
                                                {
                                                cout<<"\nPlease ensure digits are in the correct place.";
                                                }
                                }//endlength check
                                else
                                        {
                                        cout<<"\n Please ensure the postcode is of a suitable length.";
                                        }
                                }//end if - length
        }//endelse for length is 7

return valid;
}//endofpostcodecheck

//Validation routine to check that the mobile number entered begins '07'
int TelValCust(char phonenum[12])
{
//local variables
```

```
int valid = 0;
int position;
int len=strlen(phonenum);

if(len==0)
        {
        valid=0;
        return valid;
        }

if(phonenum[0]=='0')     //Checks first digit is 0
        {
        if(phonenum[1]=='7')    //Checks second digit is 7
                {
                //Loops through the rest of the characters entered
                for(position=2;position<len;position++)
                        {
                        //Checks rest of the characters are numbers
                        if(isdigit(phonenum[position]))
                                {
                                valid = 1;
                                }//endif for position 10
                                else
                                        {
                                        //Error message if the number entered is not all digits
                                        cout<<"\nPlease ensure the mobile number is all digits.";
                                        valid=0;
                                        return valid;
                                        }//end else
                        }//endfor - checking each character is an integer
                }//endif - checking second digit is a 7
                else
                        {
                        //Error message is outputted to make sure mobile is in the correct format
                        cout<<"\nPlease make sure the phone number begins 07.";
                        }//end else
        }//end if - checking first digit is 0
        else
                {
                //Error message is outputted to make sure mobile is in the correct format
                cout<<"\nPlease enter a phone number beginning with 07";
                }//endelse

return valid;
}

//This function determines whether or not the change customer menu should be outputted to the screen
int ChangeCustomer()
{
int changestatus=0;
//While loop to output the customer menu
while(changestatus != 3)
        {
        changestatus = ChangeCustomerMenu();
        }//endwhile
```

```
getch();
return 0;
}

//Routine that allows a user to enter different functions for changing information about a customer
int ChangeCustomerMenu()
{
int changecustomerchoice;
//Outputs change customer menu options to the user and allows them to choose one of them
cout<<"\n \t Change Customer Information";
cout<<"\n \t ************************\n \n";
cout<<"\n \t 1. Change home address";
cout<<"\n \t 2. Change telephone number";
cout<<"\n \t 3. Return to Customer Menu";
cout<<"\n";
cout<<"\n \t Enter choice: ";
getch();
cin>>changecustomerchoice;   //User enters their choice on the menu
clrscr();
switch(changecustomerchoice) //Allows different functions to be run depending on the menu option entered by the user
        {
        case 1:  //If changecustomerchoice =1 then the CHomeAdd function is ran
             {
             CHomeAdd();
             break;
             }
        case 2:   //If changecustomerchoice =2 then the CTelNum function is ran
             {
             CTelNum();
             break;
             }

        case 3: //If changecustomerchoice =3 then it returns to the customer menu
             {
             break;
             }
        default:
               {
               //If an incorrect number is entered the following error message is ouputted
               cout<<"\nPlease enter a number between 1 and 3.";
               getch();
               break;
               }
        }//endcase

clrscr();
getch();
return changecustomerchoice;
}

//Allows a user to change the home address of a customer to store in the customer file
int CHomeAdd()
{
//local variables
char name[15];
```

```
int compare;
int compresult;
char result[2];
int adone=0;
int adthree=0;
int post=0;
int len;

cout<<"\nChange Customer Home Address";
cout<<"\n**************************";

cin.get();
cout<<"\nEnter last name of the customer: ";
cin.getline(name,15);   //Customer to be searched for
len=strlen(name); //Extracts the number of characters and stores it under the variable 'len'
if(len==0)   //Checks to see if data has been entered
        {
        //If no data has been entered then the following error message is outputted
        cout<<"\nPlease ensure a last name has been entered.";
        getch();
        return 0;
        }

for(custref=0;custref<10;custref++)  //loops through all the customerss in the file
        {
        ifstream fin(FileName2,ios::binary);     //Calculation to find customer records in the file
        fin.seekg(custref*sizeof(a_cust));
        fin.get((char*)&a_cust,sizeof(a_cust));
        fin.close();
        compare=strcmpi(a_cust.lnamecust,name);   //compares last name entered with those in the customer file
        if(compare==0)  //If compare==0, then a match has been found between the last names in the customer file and the one entered by the
                {
                //Details about the customer are outputted to check its the correct customer
                cout<<"\nFull name: "<<a_cust.fnamecust<<" "<<a_cust.lnamecust;
                cout<<"\n";
                cout<<"\nCurrent address: "<<a_cust.oneadcust;;
                cout<<"\n\t\t "<<a_cust.twoadcust;
                cout<<"\n\t\t "<<a_cust.threeadcust;
                cout<<"\n\t\t "<<a_cust.pcodecust;
                cin.get();
                cout<<"\nIs this the correct customer to be amended (Y/N): ";
                cin>>result; //User confirms whether or not is it the correct customer
                compresult=strcmpi(result,"Y");   //Compares the input with Y
                if(compresult == 0)     //If the input is Y then the new address can be added, if it is N returns to the Change Customer Men
                        {
                        ofstream fout(FileName2,ios::in);
                        fout.seekp(custref*sizeof(a_cust));
                        cin.get();
                        //Validates the new address upon entry
                        while(adone==0)  //While loop that controls the validation to ensure address line 1 is entered
                                {
                                cout<<"\nEnter address line 1: ";
                                cin.getline(a_cust.oneadcust,30);
                                adone = PresValOne(a_cust.oneadcust);
                                }// end while
```

```
                        cout<<"\nEnter address line 2: ";  //No validation on address line 2 as it is not mandatory for an address
                        cin.getline(a_cust.twoadcust,30);

                    while(adthree==0)    //While loop that controls the validation to ensure address line 3 is entered
                        {
                        cout<<"\nEnter address line 3: ";
                        cin.getline(a_cust.threeadcust,15);
                        adthree = PresValThree(a_cust.threeadcust);
                        }// end while
                    while(post==0)       //While loop that controls validation to ensure the postcode entered is a valid UK postcode
                        {
                        cout<<"\nEnter postcode: ";
                        cin.getline(a_cust.pcodecust,9);
                        post = PostcodeValCust(a_cust.pcodecust);
                        }// end while

                    strcpy(a_cust.flag,"1");    //Changes the value of the flag to 1 to show that information is stored here
                    fout.write((char*)&a_cust,sizeof(a_cust));   //New address is written to the file
                    fout.close();
                    return 0;
                    }//end if
                    return 0;
                }//end if
        }//endfor

if(compare!=0)
        {
        //If no match is found between the last name entered by the user and those in the file the following error essage is outputted to th
        cout<<"\nCustomer's last name has not been found.";
        getch();
        return 0;
        }//end if
getch();
return 0;
}

//Validation routine to check that the mobile number entered begins '07'
int CTelNum()
{
//local variables
char name[15];
int compare;
int compresult;
char result[2];
int tel=0;
int len;

cout<<"\nChange Customer Mobile Number";
cout<<"\n*****************************";

cin.get();
cout<<"\nEnter last name of the customer: ";
cin.getline(name,15);    //Customer to be found
len=strlen(name); //Extracts number of characters entered and stores it under the variable 'len'
```

```
if(len==0) //Checks if data has been entered
        {
        //If no data has been entered then the following error message is outputted
        cout<<"\nPlease ensure a last name has been entered.";
        getch();
        return 0;
        }

for(custref=0;custref<10;custref++)    //Searches through all the customers in the file
        {
        ifstream fin(FileName2,ios::binary);
        fin.seekg(custref*sizeof(a_cust));       //Calculation to find customer records in the file
        fin.get((char*)&a_cust,sizeof(a_cust));
        fin.close();
        compare=strcmpi(a_cust.lnamecust,name);  //Compares each customer's last name in the file with the one entered
        if(compare==0)   //If compare==0, then a match has been found between the last names in the customer file and the one entered by the
                {
                //Details about the customer are outputted to check its the right customer
                cout<<"\nFull name: "<<a_cust.fnamecust<<" "<<a_cust.lnamecust;
                cout<<"\nMobile number: "<<a_cust.telnocust;
                cin.get();
                cout<<"\nIs this the correct customer to be amended (Y/N): ";
                cin>>result; //User confirms whether or not is it the correct customer
                compresult=strcmpi(result,"Y");  //Compares the input with Y
                if(compresult == 0) //If the input is Y then the new mobile number can be added, if it is N returns to the Change Customer M
                        {
                        ofstream fout(FileName2,ios::in);
                        fout.seekp(custref*sizeof(a_cust));
                        cin.get();
                        //Validates the new mobile number upon entry
                        while(tel==0)    //While loop that controls the validation to ensure the mobile number entered is of the correct for
                            {
                            cout<<"\nEnter mobile number: ";
                            cin.getline(a_cust.telnocust,12);
                            tel = TelValCust(a_cust.telnocust);
                            }// end while

                        strcpy(a_cust.flag,"1"); //Changes the value of the flag to 1 to show that information is stored here
                        fout.write((char*)&a_cust,sizeof(a_cust));   //Writes new number to the file
                        fout.close();
                        return 0;
                        }//end if
                        return 0;
                }//end if
        }//endfor
if(compare!=0)
        {
        //If no match is found between the last name entered by the user and those in the file the following error essage is outputted to th
        cout<<"\nCustomer's last name has not been found.";
        getch();
        return 0;
        }//end if
getch();
return 0;
}
```

```
//Allows the user to delete a customer from the customer file
int DeleteCustomer()
{
//local variables
int compare;
int compresult;
char result[2];
int len;
char charref[3];

cout<<"\nDelete a customer";
cout<<"\n*****************";

cout<<"\nEnter customer reference: ";
cin.get();
cin.getline(charref,3);       //Customer reference to be found

len=strlen(charref);  //Used to extract the number of characters and stores this valid under len

if(len==0) //Checks to see if data has been entered
        {
        //If no data has been entered then the following error message is outputted
        cout<<"\nPlease ensure a customer reference has been entered.";
        getch();
        return 0;
        }//end if
else
        {
        custref=atoi(charref);
        }//end else

ScheduleCheck(custref);     //Check to see if the customer has a booking on the schedule

ifstream fin(FileName2,ios::binary);
fin.seekg(custref*sizeof(a_cust));     //Calculation to find customer records in the file
fin.get((char*)&a_cust,sizeof(a_cust));
fin.close();
compare = strcmpi(a_cust.flag,"1");    //Checks to see if there is actually data stored at this location
if(compare==0)   //Compare==0 if there is data stored at that location in the file
        {
        //Details about the customer are outputted to check its the right customer
        cout<<"\nFull name: "<<a_cust.fnamecust<<" "<<a_cust.lnamecust;
        cout<<"\nMobile number: "<<a_cust.telnocust;
        cin.get();
        cout<<"\nIs this the correct customer to be deleted (Y/N): ";
        cin>>result;   //User confirms whether or not is it the correct customer
        compresult=strcmpi(result,"Y");  //Compares the input with Y
        if(compresult == 0)  //If the input is Y then the customer information can be deleted, if it is N returns to the Customer Menu
                {
                ofstream fout(FileName2,ios::in);
                fout.seekp(custref*sizeof(a_cust));
                strcpy(a_cust.flag,"0");       //Location is now shown as empty, more data can then be written here
                fout.write((char*)&a_cust,sizeof(a_cust));
                fout.close();
```

```
                }// end if
                return 0;
        }//end if
if(compare!=0)
        {
        //If customer reference's location is empty - outputs the following error message
        cout<<"\nCustomer reference not found.";
        getch();
        return 0;
        }//end if
getch();
return 0;
}

//Routine that checks if the customer to be deleted has a future booking recorded on the schedule
int ScheduleCheck(int cust)
{
//local variables
int find;
int compare;
char custchar[3];
itoa(cust, custchar,10);   //converts integer for customer reference into a character

ReadBackQuotesFile();
ReadBackScheduleFile();

for(find=0;find<nqi;find++)   //Searches through quotes stored in the quote file
        {
        compare = strcmpi(custchar,custno[find]);   //Compares customere references in the quotes file with the one entered
        if(compare==0)    //Compare==0 if there is a match between the reference entered and those in the file
                {
                if(strcmpi(quoteref[find],booking[staff][date][hour])==0);   //Searches the schedule for this quote reference
                        {
                        //Outputs this message if the quote is found
                        cout<<"\nQuote reference linked to this customer is on the schedule - are you sure you wish to delete this customer?
                        }//end if
                }//end if
        }//end for
return 0;
}

//This function determines whether or not the view customer menu should be outputted to the screen
int ViewCustomer()
{
int viewstatus=0;
//While loop to output the view customer menu
while(viewstatus != 3)
        {
        viewstatus = ViewCustomerMenu();
        }//endwhile
getch();
return 0;
}

//Outputs the customer menu to the screen to allow a user to chose an option
```

```
int ViewCustomerMenu()
{
int viewcustomerchoice;
//Outputs the view customer menu
cout<<"\n \t View Customer Information";
cout<<"\n \t ************************\n \n";
cout<<"\n \t 1. Search customer by name";
cout<<"\n \t 2. Search customer by customer reference";
cout<<"\n \t 3. Return to Customer Menu";
cout<<"\n";
cout<<"\n \t Enter choice: ";
getch();
cin>>viewcustomerchoice;  //User input for menu choice
clrscr();
switch(viewcustomerchoice) //Allows different menu options to be ran depending on the menu option entered by the user
        {
        case 1:  //If viewcustomerchoice=1 then the CName unction is ran
             {
             CName();
             break;
             }
        case 2: //If viewcustomerchoice=2 then the CRef function is ran
             {
             CRef();
             break;
             }

        case 3:  //If viewcustomerchoice=3 then it returns to the customer menu
             {
             break;
             }
        default:
               {
                //If user enters a wrong number outside the ramge this error message is outputted
               cout<<"\nPlease enter a number between 1 and 3.";
               getch();
               break;
               }
        }//endcase

clrscr();
getch();
return viewcustomerchoice;
}

//Allows the user to view a customer by the customers last name
int CName()
{
//local variables
int compare;
int cmp;
char name[25];
int len;

cout<<"\nView Customer by Customer Name";
```

```
cout<<"\n*****************************";

cout<<"\nEnter customers last name: ";
cin.get();
cin.getline(name,25);  //Customer to be found

len=strlen(name); //Extracts number of characters entered and stores it under the variable 'len'
if(len==0) //Checks if data has been entered
        {
        //If no data has been entered then the following error message is outputted
        cout<<"\nPlease ensure a last name has been entered.";
        getch();
        return 0;
        }
for(custref=0;custref<10;custref++)  //Searches through all customers in the file
        {
        ifstream fin(FileName2,ios::binary);
        fin.seekg(custref*sizeof(a_cust));       //Calculation to find customer records in the file
        fin.get((char*)&a_cust,sizeof(a_cust));
        fin.close();
        compare=strcmpi(a_cust.lnamecust, name);    //Compares each last name in the file with the one entered
        if(compare == 0)   //Compare==0 if there is a match between the last name entered and one stored in the file
                {
                cmp = strcmpi(a_cust.flag, "0");      //Checks to see if there is no data stored
                        if(cmp!=0)     //If there is data stored in the file, information is outputted to the screen
                                {
                                cout<<"\nFull name: "<<a_cust.title<<" "<<a_cust.fnamecust <<" "<<a_cust.lnamecust;
                                cout<<"\n";
                                cout<<"\nAddress line 1: "<<a_cust.oneadcust;
                                cout<<"\nAddress line 2: "<<a_cust.twoadcust;
                                cout<<"\nAddress line 3: "<<a_cust.threeadcust;
                                cout<<"\nPostcode: "<<a_cust.pcodecust;
                                cout<<"\n";
                                cout<<"\Mobile number: "<<a_cust.telnocust;
                                getch();
                                return 0;
                                }// end if
                }//end if
        }//end for
if(compare!=0)
        {
        //If no match is found between the last name entered by the user and those in the file the following error essage is outputted to th
        cout<<"\nCustomer's last name has not been found.";
        getch();
        return 0;
        }//end if
getch();
return 0;
}

//User enters customer refernce to view all information stored on that customer
int CRef()
{
int compare;
char charref[3];
```

```
int len;


cout<<"\nView Customer by Customer Reference";
cout<<"\n*****************************";

cout<<"\nEnter customer reference: ";
cin.get();
cin.getline(charref,3);        //Customer reference to be searched
len=strlen(charref);  //Used to extract the number of characters and stores this valid under len

if(len==0) //Checks to see if data has been entered
        {
        //If no data has been entered then the following error message is outputted
        cout<<"\nPlease ensure a customer reference has been entered.";
        getch();
        return 0;
        }//end if
else
        {
        custref=atoi(charref);
        }//end else
ifstream fin(FileName2,ios::binary);
fin.seekg(custref*sizeof(a_cust));      //Calculation to find customer records in the file
fin.get((char*)&a_cust,sizeof(a_cust));
fin.close();
compare=strcmpi(a_cust.flag, "0");     //Checks to see if there is no data at this location
if(compare != 0)       //If compare!=0 this means that data is stored at the location
        {
        //Customer information is outputted to the screen
        cout<<"\nFull name: "<<a_cust.title<<" "<<a_cust.fnamecust <<" "<<a_cust.lnamecust;
        cout<<"\n";
        cout<<"\nAddress line 1: "<<a_cust.oneadcust;
        cout<<"\nAddress line 2: "<<a_cust.twoadcust;
        cout<<"\nAddress line 3: "<<a_cust.threeadcust;
        cout<<"\nPostcode: "<<a_cust.pcodecust;
        cout<<"\n";
        cout<<"\nMobile number: "<<a_cust.telnocust;
        getch();
        return 0;
        }// end if

if(compare==0)
        {
        //If flag location is 0 , meaning its empty and no information is stored there - the following error message is outputtea
        cout<<"\nCustomer reference not found.";
        getch();
        return 0;
        }//end if
getch();
return 0;
}

//This function determines whether or not the quote menu should be outputted to the screen
int Quotes()
```

```
{
int quotestatus=0;
//While loop to output the quote menu
while(quotestatus != 5)
        {
        quotestatus = QuoteMenu();
        }//endwhile
getch();
return 0;
}

//Outputs the quote menu to the screen allowing the user to access different parts of the system regarding quotes
int QuoteMenu()
{
int quotechoice;
//Quote menu is outputted to the screen
cout<<"\n \t Quotes Menu";
cout<<"\n \t ***********\n \n";
cout<<"\n \t 1. Add Quote";
cout<<"\n \t 2. Change quote information";
cout<<"\n \t 3. Delete quote";
cout<<"\n \t 4. View quote";
cout<<"\n \t 5. Return to Main Menu";
cout<<"\n";
cout<<"\n \t Enter choice: ";
getch();
cin>>quotechoice;  //Input from user for what they wish to access
clrscr();
switch(quotechoice)   //Allows different functions to be ran deoening on the menu option entered by the user
        {
        case 1: //If quotechoice=1 then the AddQuote function is ran
             {
             AddQuote();
             break;
             }
        case 2:  //If quotechoice=1 then the ChangeQuote function is ran
             {
             if(level>=2)
                {
                ChangeQuote();
                }
             else
                {
                //Error message is outputtef if the user logged in does not have a high enough access level
                cout<<"\nNot authorised.";
                getch();
                }
             break;
             }

        case 3:  //If quotechoice=3 then the DeleteQuote function is ran
             {
             if(level>=2)  //Only allows access to the staff with level of acess of 2 or 3
                {
                DeleteQuote();
```

```
            }
         else
            {
            //Error message is outputtef if the user logged in does not have a high enough access level
            cout<<"\nNot authorised.";
            getch();
            }
         break;
         }

      case 4: //If quotechoice=3 then the ViewQuote function is ran
         {
         ViewQuote();
         break;
         }

      case 5:  //If quotechoice=5 then it returns to the MainMenu
         {
         break;
         }
      default:
           {
           //If user enteres a digit outside of the range - following error message is outputted
           cout<<"\nPlease enter a number between 1 and 5.";
           getch();
           break;
           }
      }//endcase


clrscr();
getch();
return quotechoice;
}

//Allows the user to add a quote, automatically linking the prices of materials from the stock file
int AddQuote()
{
//local variables
int numofitems;
int count;
int quotenum;
int stocktotal =0;

int numofdaysq;
int mileageq;
int VAT;
int labour;
int total;

int qref=0;
int cust=0;
int dateq=0;
int presm=0;
int dayrange=0;
```

```
int prestravel=0;


int datebuff=0;
char datein[30];

cout<<"\nAdd Quote";
cout<<"\n*********";
ReadBackQuotesFile();
cin.get();
//Data added regarding a quote is validated upon entry
while(qref == 0)         //While loop that controls validation to ensure that the quote reference entered isn't already in use
        {
        cout<<"\nEnter quote reference: ";
        cin.getline(quoteref[nqi],3);
        qref = UniqueQ(quoteref[nqi]);
        }//end while

while(cust==0)  //While loop that controls validation to ensure that the customer reference entered exists
        {
        cout<<"\nEnter customer reference: ";
        cin.getline(custno[nqi],3);
        cust = LocateCustQVal(custno[nqi]);
        }//end while

while(dateq == 0 || datebuff==0)   //While loop that controls validation to ensure that the date entered is in the correct format: DD/MM/YYY
        {
        cout<<"\nEnter date quote is produced: ";
        cin.getline(datein,30);
        datebuff = Buffer(datein,10);
        if(datebuff!=0 && dateq==0)
                {
                dateq = DateVal(datein);
                }//end if
        } // end while
strcpy(quotedate[nqi],datein);

while(presm == 0)    //While loop that controls validation to ensure that data has been entered
        {
        cout<<"\nEnter job description: ";
        cin.getline(mainjobdesc[nqi],50);
        presm = PresValJ(mainjobdesc[nqi]);
        }// end while

while(dayrange == 0)  //While loop that controls validation to ensure that the number of days enterd is within a sensible range
        {
        cout<<"\nEnter number of days on the job: ";
        cin.getline(numofdays[nqi],3);
        dayrange = RangeDay(numofdays[nqi]);
        }// end while

while(prestravel == 0)  //While loop that controls validation to ensure that the mileage costs entered are within a sensible range
        {
        cout<<"\nEnter travel costs: ";
        cin.getline(mileage[nqi],4);
```

```
        prestravel = PresValTr(mileage[nqi]);
        }

stocktotal = AddStockQuote(quoteref[nqi]);
cout<<stocktotal;

// converts values entered by the user to integers to be used in the calculation
numofdaysq = atoi(numofdays[nqi]);
mileageq = atoi(mileage[nqi]);

//Calculates the whole cost of the quote
VAT = 0.2 * (stocktotal + (numofdaysq*150)+ mileageq);
labour = (numofdaysq*150);
total = stocktotal + (numofdaysq*150) + mileageq + VAT;

//Convers the results from the calculation so it can be stored in the quotes file
itoa(labour,labourq[nqi],10);
itoa(stocktotal,stockcost[nqi],10);
cout<<"\nchar"<<stockcost[nqi];
itoa(total,totalcost[nqi],10);
cout<<"\nchar"<<totalcost[nqi];
itoa(VAT, vat[nqi],10);

cout<<"\nPrice Breakdown";         //Outputs all the price calculations to the screen once all the information has been entered
cout<<"\n***************";
cout<<"\nMaterials: "<<stocktotal;
cout<<"\nLabour: "<<(numofdaysq*150);
cout<<"\nMileage: "<<mileageq;
cout<<"\nVAT: "<<VAT;
cout<<"\nTotal: "<<total;

nqi = nqi +1;     //Increases the number of quotes stored in the file by 1
itoa(nqi,nqc,10);
ReWriteQuotesFile();   //Data is written to the file
getch();
return 0;
}

int AddStockQuote(char quote[2])
{
int count;
int cost=0;
int stocktotal=0;
int quotenum=atoi(quote);
char sref[2];

cout<<"\nHow many items of stock are required: ";
cin>>numofitems;
for(count=0;count<numofitems;count++)     //for the number of items required finds the stock price and adds it to a running total
        {
        ReadBackLinksFile();
        //save the quote reference entered by the user to the links file
        quotenum = atoi(quoteref[nqi]);
        itoa(quotenum,linksquoteref[nli],10);
```

```
        //User inputs the stock reference which is then passed a parameter to find in the stock file and retrieve the price
        while(stock==0)
                {
                cout<<"\nPlease enter the stock reference: ";
                cin>>sref;
                cin.get();
                cin.getline(linksstockref[nli],3);
                stock = StockVal(sref);
                if(stock!=0)
                        {
                        nli = nli +1;
                        ReWriteLinksFile();
                        cost = FindStock(sref,count);
                        //Calculates the total cost of all stock
                        stocktotal = cost + stocktotal;
                        count=count+1;
                        }

                }



        }
getch();
return stocktotal;
}

//Routine which validates the stock reference for add quote
int StockVal(char ref[2])
{
//local variables
int find;
int len;
int compare;
int valid=0;
int position;

len=strlen(ref);   //Extracts number of characters entered

if(len==0)  //If len==0 no data has been entered
        {
        //Error message is outputted if no data has been entered
        cout<<"\nPlease ensure you have entered a stock reference.";
        getch();
        valid=0;
        }

for(position=0;position<len;position++)  //Loops through all characters
        {
        //Checks its a number
        if(isdigit(ref[position]))
                {
                ReadBackStockFile();
                //Loops through all stock in the file
                for(find=0;find<nsti;find++)
```

```
                        {
                        //Compares reference entered with those stored in the file
                        compare=strcmpi(ref, stockref[find]);
                        //If there is a match, reference is accepted
                        if(compare==0)
                                {
                                valid=1;
                                return valid;
                                }//end if
                        }//end for
                }//end if
                else
                        {
                        //Error message if a number is not entered
                        cout<<"\nPlease ensure a number is entered.";
                        getch();
                        valid=0;
                        return valid;
                        }//end else
                }//endfor
if(compare!=0)
        {
        //Error message is outputted if no match is found
        cout<<"\nStock reference not found. Please try again.";
        }
return valid;
}

//Uses the stock reference entered by the user to find the price of stock and creates a running total of all the stock required to pass back
int FindStock(char lstock[3], int counting)
{

ReadBackStockFile();
//local variables
int find;
int compare;
int cost=0;
int quant;
int sp; // sp = stock price
ReadBackStockFile();
for(find=0;find<nsti;find++)              //Searches through all stock stored in the file
        {
        compare = strcmpi(lstock,stockref[find]);    //Compares stock references in the file with the one entered
        if(compare==0)
                {
                                        //If a match is found, price of materials for the quote can be calculated
                sp = atoi(stockprice[find]);              //converts price of stock to an integer to be used in calculations
                cout<<"\nWhat quantity is required: ";
                cin>>quant;
                counting=counting+1;
                int fileq = atoi(quantity[find]);   //Converts quantity of item of stock stored in the file to an integer
                fileq= fileq - quant;       //Takes off the quantity required for this quote
                itoa(fileq,quantity[find],10);
                ReWriteStockFile();            //Saves new quantity to the stock file
                itoa(quant,linksquantity[nli],10);
```

```
                        cost = cost + (sp*quant);          //Calculates the price of 1 item of stock and multiplies it by the quantity requi
                        if(counting<numofitems)    //If stock reference still left to be validated
                                {
                                stock=0;
                                }
                                        //Adds it to a running total

                }//end if
        }//end for
getch();
return cost;   //Returns the final price of all the materials to the AddQuote function to be added to the overall price
}

//Validation routine to ensure the quote reference entered is unique
int UniqueQ(char quote[3])
{
//local variables
int uniqueref=1;
int find;
int compare;
int len;
len=strlen(quote);
int position;

ReadBackQuotesFile();

if(len!=0)     //Checks something has been entered
        {
        for(position=0;position<len;position++)     //loops through all characters in inout
                {
                if(isdigit(quote[position]))      //Checks to see if it an integer, if it is the rest of the loop runs
                {
                for (find = 0; find < nqi; find++)      //Searches through all the quotes in the file
                        {
                        compare = strcmpi(quoteref[find], quote);    //Compares quote references in the file with the one entered
                        if (compare == 0)
                                {                                       //If a match is found, quote reference is already in use
                                uniqueref = 0;
                                cout<<"\nQuote reference already in use. Please try again.";
                                } //endif found record
                        }//endfor
                }//endif
                else
                        {
                        cout<<"\nPlease ensure a number was entered for the reference.";
                        uniqueref=0;
                        getch();
                        return uniqueref;
                        } //end else
                }//end for
        }//end if
        else
                {
                //If nothing is entered following error message is outputted
                cout<<"\nPlease ensure a quote reference was entered.";
```

```
                uniqueref=0;
                }//end else
return uniqueref;
}

//Validation routine to check the customer reference entered exists
int LocateCustQVal(char custno[3])
{
//local variables
int compare;
int valid =0;

sscanf(&custno[0],"%d",& custref);
ifstream fin(FileName2,ios::binary);        //Calculation to jump the right position in file using the reference entered by the user
fin.seekg(custref*sizeof(a_cust));
fin.get((char*)&a_cust,sizeof(a_cust));
fin.close();

compare=strcmpi(a_cust.flag,"1");   // checks if there is data stored at that location
if(compare==0)                      //If there is, customer reference exists
        {
        valid=1;
        }//end if

if(compare!=0)
        {
        //If flag is empty, reference not found and therefore error message is outputted
        cout<<"\nCustomer reference not found.";
        }//end if
return valid;
}

//Validation routine that checks that the date entered is in the correct format  DD/MM/YYYY
int DateVal(char date[11])
{
//local variables
int valid =0;
int months;
int days;
float years;


if(date[2] == '/' && date[5] == '/')    //Ensure the forward slashes are in the correct place
        {
        sscanf(&date[3],"%d",&months);      //Extracts the days amd month from the date entered and converts them to integers
        sscanf(&date[0],"%d",&days);
        if(months==4 || months==6 || months==9 || months==11)
                {                                                   //Checks the days entered are between the 1st and 30th for these months
                if(days>= 1 && days<= 30)
                        {
                        valid = 1;
                        }//endif-fordays30
                else                                        //If not - error message is outputted
                        {
                        cout<<"\nMake sure the date is between the 1st and 30th";
```

```
                }//endelse
        }//endif-formonths30

        if(months==1 || months==3 || months==5 || months==7 || months==8 || months==10 || months==12)    //Checks the days entered are betwee
                {
                if(days>= 1 && days<= 31)
                        {
                        valid = 1;              //If not - error message is outputted
                        }//endif-fordays31
                else
                        {
                        cout<<"\nMake sure the date is between the 1st and the 31st";
                        }//endelse
                }//endif-formonths31

        if(months==2)
                {
                sscanf(&date[7],"%f",&years);    //If month is Feb converts years to float to allow for division to find if it is a leap yea
                if(floor(years/4)== (years/4))     //If division is the same - shows its a leap year
                        {
                        if(days>= 1 && days <=29)   //Allows up the 29th
                                {
                                valid = 1;
                                }//endif-fordaysfeb29
                        else
                                {
                                cout<<"\nMake sure the date is between the 1st and the 29th";
                                }
                        }

                if(floor(years/4)!= (years/4))       //If division is not equal - allows only until the 28th
                                {
                                if(days>= 1 && days <=28)
                                        {
                                        valid = 1;
                                        }//endif-fordaysfeb28
                                else
                                        {
                                        cout<<"\nMake sure the date is between the 1st and the 28th";
                                        }
                                }//endif
                }//end if - months

        }//endif-forslashes
else
        {
        cout<<"Forward slashes are in the wrong place";
        }//endelse

return valid;
}

//Validation routine to check data has been entered
int PresValJ(char presence[30])
{
```

```
int length;
length=strlen(presence);    //strlen extracts the number of characetrs entered and stores it under the variable length
if(length==0)   //Checks to see if the length is 0 as if it is no data has been entered
        {
        //Error message is outputted if length = 0
        cout<<"\nPlease enter a job description";
        }//endif
return length;
}

//Validation routine that checks that the number of days entered is reasonable
int RangeDay(char day[4])
{
//local variables
int valid = 0;
int intday;
int position;
int len;
len=strlen(day); //extracts the number of characters stored

intday = atoi(day);   //converts number of days to an integer

if(len!=0)
        {
        for(position=0;position<len;position++) //Loops through all the characters
                {
                if(isdigit(day[position]))  //Checks the input is a number
                        {
                        if(intday>=1 && intday <=21)       //Ensure that the number of days on the job is less that 3 weeks
                                {
                                valid = 1;
                                } // end if
                        else
                                {
                                //Error message if not in range
                                cout<<"\nPlease ensure the number of days entered is within the range 1-21 days.";
                                getch();
                                valid=0;
                                return valid;
                                }// end else
                        }//end if
                        else
                                {
                                //Error message outputted if not a number
                                cout<<"\nPlease enter a number.";
                                getch();
                                valid=0;
                                return valid;
                                }//end else
                }//endfor
        }//end if
        else
                {
                //Error message outputted if no data is entered
                cout<<"\nPlease ensure the number of days was entered.";
```

```
                getch();
                valid=0;
                return valid;
                }//end else
return valid;
}

//Validation routine to check data has been entered
int PresValTr(char presence[30])
{
int length;
int position;
length=strlen(presence); //strlen extracts the number of characetrs entered and stores it under the variable length

for(position=0;position<length;position++) //Loops through all the characters
                {
                if(isalpha(presence[position]))  //Checks the input is a number
                        {
                        cout<<"\nPlease ensure a number has been entered.";
                        getch();
                        length=0;
                        return length;
                        }//end if
                }//endfor

if(length==0)     //Checks to see if the length is 0 as if it is no data has been entered
        {
        //Error message is outputted if length = 0
        cout<<"Please enter travel costs";
        }//endif
return length;
}

//This function determines whether or not the change quote menu should be outputted to the screen
int ChangeQuote()
{
int quotestatus=0;
//While loop for outputting change quote options
while(quotestatus != 3)
        {
        quotestatus = ChangeQuoteMenu();
        }//endwhile
getch();
return 0;
}

//Routine that allows a user to enter different functions for changing information about a quote
int ChangeQuoteMenu()
{
int changequotechoice;
//Outputs change quote menu options to the user and allows them to choose one of them
cout<<"\n \t Change Quote Information";
cout<<"\n \t ************************\n \n";
cout<<"\n \t 1. Change quote price";
cout<<"\n \t 2. Change number of days required";
```

```
cout<<"\n \t 3. Return to Quote Menu";
cout<<"\n";
cout<<"\n \t Enter choice: ";
getch();
cin>>changequotechoice;  //inputted choice from user
clrscr();
switch(changequotechoice) //Allows different functions to be run depending on the menu option entered by the user
        {
        case 1:  //If changequotechoice=1 then the QPrice function will be ran
             {
             QPrice();
             break;
             }
        case 2:   //If changequotechoice=2 then the QDay function will be ran
             {
             QDay();
             break;
             }

        case 3:  //If changequotechoice=3 then it will return to the Quote Menu
             {
             break;
             }
        default:
               {
               //If user enters a number outside the range the following error message is outputted
               cout<<"\nPlease enter a number between 1 and 3.";
               getch();
               break;
               }
        }//endcase


clrscr();
getch();
return changequotechoice;
}

//Allows the user to change the price of the quote
int QPrice()
{
//local variables
char looking[25];
int compare;
int compresult;
int find;
char result[2];
int priceval=0;

cout<<"\nChange price of quote";
cout<<"\n**********************";
ReadBackQuotesFile();
cin.get();

cout<<"\nEnter the quote reference of the quote price you want to change: ";
```

```
cin.getline(looking,25);                        //Quote to be found
for(find =0; find<nqi; find++)                  //Searches through all the quotes in the file
        {
        compare = strcmpi(looking,quoteref[find]);   //Compares each quote reference in the file with the one entered
        if(compare ==0)   //If compare==0 then there is a match between the quote reference entered and one stored in the file
                {
                //Details about the quote are outputted to check its the correct quote
                cout<<"\nCustomer Reference: "<<custno[find];
                LocateCust(custno[find]);   //Function which links customer information to also be outputted
                cout<<"\nJob Description: "<<mainjobdesc[find];
                cin.get();
                cout<<"\nIs this the correct quote to be amended (Y/N): ";
                cin>>result;  //User confirms whether or not it is the correct quote
                compresult=strcmpi(result,"Y");   //Compares the input with Y
                if(compresult == 0)  //If the input is Y then the new price can be added, if it is N returns to the Change Quote Menu
                        {
                        cin.get();
                        //validates new price upon entry
                        while(priceval==0)   //while loop that controls the validation to ensure the price entered is within a suitable rang
                                {
                                cout<<"\nEnter new price: ";
                                cin.getline(totalcost[find],6);
                                priceval = QuotePriceRange(totalcost[find]);
                                }
                        ReWriteQuotesFile();              //New price is written to the file
                        return 0;
                        }// end if
                        return 0;
                }//end if
           }//end for
if(compare!=0)
        {
        //If there is not a match between the reference entered and those stored in the file then the following error message is outputted
        cout<<"\nQuote reference not found.";
        }//end if
getch();
return 0;
}

//Validation routine that checks the price entered is reasonable
int QuotePriceRange(char cost[6])
{
//local variables
int price;
int valid=0;
int len;
len=strlen(cost);  //strlen extracts the number of characetrs entered and stores it under the variable len

if(len==0)  //Checks to see if the length is 0 as if it is no data has been entered
        {
        //Error message is outputted if len=0
        cout<<"\nPlease ensure you have entered a new price.";
        getch();
        valid=0;
        return valid;
```

```
        }
price = atoi(cost); //converts price to an integer for range check


if(price>=50 && price<=10000)  //range check
        {
        valid=1;
        }//end if
else
        {
        //Error message if the price is outside of the range
        cout<<"\nPlease enusure the price entered is wihtin the range of 50-10000";
        }//end else
return valid;
}

//Allows the user to change the number of days on the job and therefore the labour and total cost
int QDay()
{
//local variables
char looking[25];
int compare;
int compresult;
int find;
char result[2];
int numofdaysq;
int newlabour;
int oldlabour;
int difference;
int totali;
int newtotal;
int dayrange=0;
int len;

cout<<"\nChange days worked on quote";
cout<<"\n**************************";
ReadBackQuotesFile();
cin.get();


cout<<"\nEnter the quote reference of the quote you want to change: ";
cin.getline(looking,25);              //Quote to be found

len=strlen(looking);
if(len!=0)
        {
        for(find=0; find<nqi; find++)       //Searches through all the quotes
                {
                compare = strcmpi(looking,quoteref[find]);     //Compares each quote reference in the file with the one entered
                if(compare ==0)  //If compare==0 then there is a match between the quote reference entered and one stored in the file
                        {
                        //Details of the quote are outputted to check its the correct one
                        cout<<"\nCustomer Reference: "<<custno[find];
                        LocateCust(custno[find]);   //Function to output customer information
                        cout<<"\nJob Description: "<<mainjobdesc[find];
```

```
                        cin.get();
                        cout<<"\nIs this the correct quote to be amended (Y/N): ";
                        cin>>result; //User confirms whether or not it is the correct quote
                        compresult=strcmpi(result,"Y");  //Compares the input with Y
                        if(compresult == 0)  //If the input is Y then the new number of days can be added, if it is N returns to the Change
                                {
                                cin.get();
                                //Validates upon entry
                                while(dayrange==0)   //While loop that controls the validation to ensure the number of days entered is withi
                                        {
                                        cout<<"\nEnter new number of days: ";
                                        cin.getline(numofdays[find],3);
                                        dayrange = RangeDay(numofdays[find]);
                                        }//end while

                                ReWriteQuotesFile();  //saves data to the quotes file

                                //converts to integers to be used in calculations
                                numofdaysq = atoi(numofdays[find]);
                                oldlabour = atoi(labourq[find]);

                                //calculates new labour price
                                newlabour = numofdaysq*150;

                                //stores new labour price to quotes file
                                itoa(newlabour,labourq[find],10);
                                ReWriteQuotesFile();

                                //works out the extra cost for days so it can be added to the total
                                difference = newlabour - oldlabour;

                                //converts original total from quotes file to an integer to be used in calculations
                                totali = atoi(totalcost[find]);

                                //calculates new total
                                newtotal = totali + difference;

                                //stores new total cost to quotes file
                                itoa(newtotal,totalcost[find],10);
                                ReWriteQuotesFile();  //Saves new total price to the file
                                return 0;
                                }// end if
                                return 0;
                        }//end if

                   }//end for
        }//end if
        else
                {
                cout<<"\nPlease ensure a reference has been entered.";
                getch();
                return 0;
                }
if(compare!=0)
        {
```

```
        //If there is not a match between the reference entered and those stored in the file then the following error message is outputtea
        cout<<"\nQuote reference not found.";
        }//end if
getch();
return 0;
}

//Allows the user to delete a quote from the quotes file
int DeleteQuote()
{
//local variables
int find;
int compare;
int compresult;
int del;
char looking[25];
char result[2];
int len;

cout<<"\nDelete quote";
cout<<"\n************";
ReadBackQuotesFile();
cin.get();
cout<<"\nEnter the quote reference of the quote you wish to delete: ";
cin.getline(looking,25);      //quote to be found

len=strlen(looking);
if(len!=0)
        {
        for(find=0;find<nqi;find++)   //searches through all the quotes in the file
                {
                compare = strcmpi (looking,quoteref[find]);        //compares each quote reference in the file with the one entered
                if(compare==0)   //If compatr ==0 then there is a match between the quote reference entered and one in the file
                        {
                        //Outputs information regarding that quote to check its the correct to be deleted
                        cout<<"\nCustomer Reference: "<<custno[find];
                        LocateCust(custno[find]);  //Functions which finds customer information to also be outputted
                        cout<<"\nJob Description: "<<mainjobdesc[find];
                        cin.get();
                        cout<<"\nIs this the correct quote to be deleted (Y/N): ";
                        cin>>result;    //User confirms whether or not it is the correct quote
                        compresult=strcmpi(result,"Y");   //Compares the input with Y
                        if(compresult == 0)  //If the input is Y then the new number of days can be added, if it is N returns to the Quote M
                                {
                                //Loops through and deletes all information for that quote
                                for(del=find;del<nqi;del++)
                                        {
                                        strcpy(quoteref[del], quoteref[del+1]);
                                        strcpy(custno[del], custno[del+1]);
                                        strcpy(quotedate[del], quotedate[del+1]);
                                        strcpy(mainjobdesc[del], mainjobdesc[del+1]);
                                        strcpy(numofdays[del], numofdays[del+1]);
                                        strcpy(labourq[del],labourq[del+1]);
                                        strcpy(mileage[del], mileage[del+1]);
                                        strcpy(vat[del], vat[del+1]);
```

```
                                        strcpy(stockcost[del],stockcost[del+1]);
                                        strcpy(totalcost[del], totalcost[del+1]);
                                        nqi=nqi -1;         //Decreases the number of quotes in the file by one
                                        itoa(nqi,nqc,10);
                                        ReWriteQuotesFile();  //Data is written to the file
                                        return 0;
                                        }// end for
                                }//end if
                                return 0;
                        }//end if
                }//end for
        }//end if
        else
                {
                cout<<"\nPlease ensure a quote reference is entered.";
                }
if(compare!=0)
        {
        //If there is not a match between the reference entered and those stored in the file then the following error message is outputtea
        cout<<"\nQuote reference not found.";
        }//end if
getch();
return 0;
}
//This function determines whether or not the view quote menu should be outputted to the screen
int ViewQuote()
{
int viewstatus=0;
//While loop for outputting quote view menu
while(viewstatus != 3)
        {
        viewstatus = ViewQuoteMenu();
        }//endwhile
getch();
return 0;
}

//Outputs view quote menu options to the user and allows them to choose one of them to run a specific function
int ViewQuoteMenu()
{
int viewquotechoice;
//Outputs the view quote menu to the screen
cout<<"\n \t View Quotes";
cout<<"\n \t ***********\n \n";
cout<<"\n \t 1. Search quote by quote reference";
cout<<"\n \t 2. Search quote by customer reference";
cout<<"\n \t 3. Return to Quote Menu";
cout<<"\n";
cout<<"\n \t Enter choice: ";
getch();
cin>>viewquotechoice;    //user input for what they want to do
clrscr();
switch(viewquotechoice)   //Allows different menu options to be ran depending on the menu option entered by the user
        {
        case 1:  //If viewquotechoice=1 then the QRef function is ran
```

```
            {
            QRef();
            break;
            }
      case 2:  //If viewquotechoice=2 then the QCustRef function is ran
            {
            QCustRef();
            break;
            }

      case 3:  //If viewquotechoice=1 then returns to the quote menu
            {
            break;
            }
      default:
              {
              //If user inputs an incorrect digit following error message is outputted
              cout<<"\nPlease enter a number between 1 and 3.";
              getch();
              break;
              }
      }//endcase


clrscr();
getch();
return viewquotechoice;
}

//Allows the user to view all the quote information by entering the quote reference
int QRef()
{
//local variables
char looking[25];
int compare;
int find;
char pound = 156;
int len;

cout<<"\nView Quote by Quote Reference";
cout<<"\n*****************************";
ReadBackQuotesFile();
cin.get();

cout<<"\nEnter quote reference: ";
cin.getline(looking,25);                  //Quote reference to be found
len=strlen(looking);
if(len!=0)
      {
      for(find=0;find<nqi;find++)              //Searches through all the quotes in the file
            {
            compare = strcmpi(looking,quoteref[find]);      //Compares the quote references in the file with the one entered
            if(compare == 0)   //If compare==0 then there is a match between the quote reference entered and one stored in the file
                  {
                  //Quote information is outputted to the screen
```

```
                        cout<<"\nCustomer reference: "<<custno[find];
                        LocateCust(custno[find]);   //Finds customer information to output with quote
                        cout<<"\n";
                        cout<<"\nJob Description: "<<mainjobdesc[find];
                        cout<<"\n";
                        cout<<"\nNumber of days worked: "<<numofdays[find];
                        cout<<"\n";
                        cout<<"\nPrice Breakdown";
                        cout<<"\n***************";
                        cout<<"\nMaterials:\t"<<pound<<stockcost[find];
                        cout<<"\nLabour: \t"<<pound<<labourq[find];
                        cout<<"\nMileage:\t"<<pound<<mileage[find];
                        cout<<"\nTotal Cost:\t"<<pound<<totalcost[find];
                        }// end if
                }// end for
        }//end if
        else
                {
                cout<<"\nPlease ensure a quote reference was entered.";
                getch();
                return 0;
                }//end else
if(compare!=0)
        {
        //If there is not a match between the reference entered and those stored in the file then the following error message is outputtea
        cout<<"\nQuote reference not found.";
        }//end if
getch();
return 0;
}

//Uses the customer reference to link customer information to output when viewing a quote
int LocateCust(char custno[3])
{
int compare;

sscanf(&custno[0],"%d",& custref);        //Calculation to locate the customer in the file
ifstream fin(FileName2,ios::binary);
fin.seekg(custref*sizeof(a_cust));
fin.get((char*)&a_cust,sizeof(a_cust));
fin.close();

compare=strcmpi(a_cust.flag,"0");  //Checks to see if the location in the file is empty
if(compare!=0)
        {
        //If location is occupied, customer information is outputted
        cout<<"\nCustomer Name: "<< a_cust.fnamecust<<" "<<a_cust.lnamecust;
        cout<<"\nCustomer Mobile Number: "<<a_cust.telnocust;
        getch();
        }//endif

if(compare==0)
        {
         //If flag is 0 so when the location in the file is empty - following error message is outputted
        cout<<"\nCustomer reference not found.";
```

```
        }// end if
getch();
return 0;
}

//Allows the user to search for a quote using the customer reference
int QCustRef()
{
//local variables
int compare;
int find;
char looking[25];
char pound = 156;
int len;

cout<<"\nView by customer reference";
cout<<"\n*************************";

ReadBackQuotesFile();
cin.get();
cout<<"\nEnter customer reference: ";        //Customer reference to be found
cin.getline(looking,25);

len=strlen(looking);

if(len!=0)
        {
        for(find=0;find<nqi;find++)                //All quotes are searched
                {
                compare = strcmpi(looking,custno[find]);    //Compares the customer references stored within the quotes file with the custom
                if(compare==0)   //If compare=0 then there is a match between the customer reference entered and one stored in the file
                        {
                        //Quote and customer information is outputted to the screen
                        cout<<"\nCustomer reference: "<<custno[find];
                        LocateCust(custno[find]);     //Uses the customer reference to find customer information from the customer file
                        cout<<"\n";
                        cout<<"\nDate quote was produced: "<<quotedate[find];
                        cout<<"\nJob Description: "<<mainjobdesc[find];
                        cout<<"\n";
                        cout<<"\nNumber of days worked: "<<numofdays[find];
                        cout<<"\n";
                        cout<<"\nPrice Breakdown";
                        cout<<"\n***************";
                        cout<<"\nMaterials:\t"<<pound<<stockcost[find];
                        cout<<"\nLabour: \t"<<pound<<labourq[find];
                        cout<<"\nMileage:\t"<<pound<<mileage[find];
                        cout<<"\nTotal Cost:\t"<<pound<<totalcost[find];
                        getch();
                        return 0;
                        } // end if
                }// end for
        }//end if
        else
                {
                cout<<"\nPlease ensure a customer reference was entered.";
```

```
                getch();
                return 0;
                }
if(compare!=0)
        {
        //If there is not a match between the reference entered and those stored in the file then the following error message is outputted
        cout<<"\nCustomer reference has not been found.";
        }//end if
getch();
return 0;
}

//Reads all quote information from the file
int ReadBackQuotesFile()
{
int count;
ifstream fin(FileName4,ios::binary);
fin.read((char*)&nqc, sizeof(nqc));
sscanf(&nqc[0], "%d", &nqi);
for(count=0;count<nqi;count++)
        {
        fin.getline(quoteref[count],sizeof(quoteref[count]));
        fin.getline(custno[count],sizeof(custno[count]));
        fin.getline(quotedate[count],sizeof(quotedate[count]));
        fin.getline(mainjobdesc[count],sizeof(mainjobdesc[count]));
        fin.getline(numofdays[count],sizeof(numofdays[count]));
        fin.getline(labourq[count],sizeof(labourq[count]));
        fin.getline(mileage[count],sizeof(mileage[count]));
        fin.getline(vat[count],sizeof(vat[count]));
        fin.getline(stockcost[count],sizeof(stockcost[count]));
        fin.getline(totalcost[count],sizeof(totalcost[count]));
        }// end for
fin.close();
return 0;
}

//Writes in new quote information to the file
int ReWriteQuotesFile()
{
int count;
ofstream fout(FileName4, ios::binary);
fout.write((char*)&nqc,sizeof(nqc));
for(count=0;count<nqi;count++)
        {
        fout.write((char*)&quoteref[count],strlen(quoteref[count]));
        fout.write("\n",1);
        fout.write((char*)&custno[count],strlen(custno[count]));
        fout.write("\n",1);
        fout.write((char*)&quotedate[count],strlen(quotedate[count]));
        fout.write("\n",1);
        fout.write((char*)&mainjobdesc[count],strlen(mainjobdesc[count]));
        fout.write("\n",1);
        fout.write((char*)&numofdays[count],strlen(numofdays[count]));
        fout.write("\n",1);
        fout.write((char*)&labourq[count],strlen(labourq[count]));
```

```
        fout.write("\n",1);
        fout.write((char*)&mileage[count],strlen(mileage[count]));
        fout.write("\n",1);
        fout.write((char*)&vat[count],strlen(vat[count]));
        fout.write("\n",1);
        fout.write((char*)&stockcost[count],strlen(stockcost[count]));
        fout.write("\n",1);
        fout.write((char*)&totalcost[count],strlen(totalcost[count]));
        fout.write("\n",1);
        }// end for
fout.close();
return 0;
}

//This function determines whether or not the invoice menu should be outputted to the screen
int Invoices()
{
int invoicestatus=0;
//While loop for outputting the invoice menu
while(invoicestatus != 5)
        {
        invoicestatus = InvoiceMenu();
        }//endwhile
getch();
return 0;
}

//Outputs the invoice menu to the screen allowing the user to access different parts of the system regarding invoices
int InvoiceMenu()
{
int invoicechoice;
//Outputs invoice menu to the screen
cout<<"\n \t Invoice Menu";
cout<<"\n \t ************\n \n";
cout<<"\n \t 1. Add Invoice";
cout<<"\n \t 2. Change invoice information";
cout<<"\n \t 3. Delete invoice";
cout<<"\n \t 4. View invoice";
cout<<"\n \t 5. Return to Main Menu";
cout<<"\n";
cout<<"\n \t Enter choice: ";
getch();
cin>>invoicechoice;   //User inputs what they want to do
clrscr();
switch(invoicechoice)  //Allows different functions to be ran deoening on the menu option entered by the user
        {
        case 1:  //If invoicechoice=1 then the AddInvoice function is ran
             {
             AddInvoice();
             break;
             }
        case 2:  //If invoicechoice=2 then the ChangeInvoice function is ran
             {
             ChangeInvoice();
             break;
```

```
        }

    case 3:  //If invoicechoice=3 then the DeleteInvoice function is ran
        {
        if(level>=2)    //The userlogged in is only granted access if they have an access level of 2 or greater
           {
           DeleteInvoice();
           }
        else
           {
           //Error message is outputted if the user does not have the correct level of access
           cout<<"\nNot authorised.";  //If access is too low this error message is outputted
           getch();
           }
        break;
        }

    case 4: //If invoicechoice=4 then the ViewInvoice function is ran
        {
        ViewInvoice();
        break;
        }

    case 5: //If invoicechoice=5 then it returns to the Main menu
        {
        break;
        }
    default:
          {
          //If a user enters an incorrect digit the following error message is outputted
          cout<<"\nPlease enter a number between 1 and 5.";
          getch();
          break;
          }
    }//endcase


clrscr();
getch();
return invoicechoice;
}

//Allows the user to add an invoice to the invoice file
int AddInvoice()
{
//local variables
int ref=0;
int cref=0;
int qref=0;
int start=0;
int endrange=0;
int end=0;
int payment=0;

char dateinstart[30];
```

```
int startbuff=0;
char dateinend[30];
int endbuff=0;
char payin[30];
int paybuff=0;

cout<<"\nAdd Invoice";
cout<<"\n***********";
ReadBackInvoiceFile();
cin.get();

//Validation routines for all information entered
while(ref==0)        //While loop that controls validation routines to ensure that the invoice reference entered is not already in use
        {
        cout<<"\nEnter invoice reference: ";
        cin.getline(invoiceref[nii],3);
        ref = UniqueInvoice(invoiceref[nii]);
        }// end while

while(cref==0)   //While loop that controls validation routines to ensure that the customer reference entered exists
        {
        cout<<"\nEnter customer reference: ";
        cin.getline(custnum[nii],3);
        cref = LocateCustIVal(custnum[nii]);
        }//end while

while(qref==0)  //While loop that controls validation routines to ensure that the quote reference entered exists
        {
        cout<<"\nEnter quote reference: ";
        cin.getline(quotenum[nii],3);
        qref= QuoteRefCheck(quotenum[nii]);
        }//end while

while(start==0 || startbuff==0)   //While loop that controls validation routines to ensure that the start date is in the correct format: DD/
        {
        cout<<"\nEnter start date: ";
        cin.getline(dateinstart,30);
        startbuff = Buffer(dateinstart,10);
        if(startbuff!=0 && start==0)
                {
                start = DateValStart(dateinstart);
                }//endif
        }// end while
strcpy(jobstartdate[nii],dateinstart);

while(end==0 || endrange==0 || endbuff==0)  //While loop that controls validation routines to ensure that the end date is in the correct for
        {
        cout<<"\nEnter end date: ";
        cin.getline(dateinend,30);
        endbuff = Buffer(dateinend,10);
        if(endbuff!=0 && (endrange==0||end==0))
                {
                end = DateValEnd(dateinend);
                endrange = RangeDate(dateinend);
                }//endif
```

```
        }// end while
strcpy(jobenddate[nii],dateinend);

while(payment ==0 || paybuff==0)   //While loop that controls validation routines to ensure that the payment status is either Y or N
        {
        cout<<"\nIs the job paid for (Y/N): ";
        cin.getline(payin,30);
        paybuff = Buffer(payin,1);
        if(paybuff!=0 && payment==0)
                {
                payment = RangePaid(payin);
                }//end if
        }// end while
strcpy(paid[nii],payin);

nii = nii +1;    //Increases the number of invoices stored in the file by one
itoa(nii,nic,10);
ReWriteInvoiceFile(); //Writes information to the file
getch();
return 0;
}

//Validation routine to check that the reference entered hasn't already been used
int UniqueInvoice(char invoice[3])
{
//local variables
int uniqueref=1;
int find;
int compare;
int len;
len=strlen(invoice);   //strlen extracts the number of characetrs entered and stores it under the variable length
int position;

ReadBackInvoiceFile();
if(len!=0)   //Checks that data had been entered
        {
        for(position=0;position<len;position++)
                {
                if(isdigit(invoice[position]))
                        {
                        for(find=0;find<nii;find++)  //loops through all the invoices file
                                {
                                compare = strcmpi(invoiceref[find], invoice);    //compares each invoice reference with the one entered
                                if (compare == 0)   //If compare==0 then there is a match between the invoice reference entered and one stor
                                        {
                                        //Error messsage is outputted to tell the user that the reference is already in use
                                        cout<<"\nInvoice reference already in use. Please try again.";
                                        uniqueref = 0;
                                        } //endif
                                }//endfor
                        }//endif
                        else
                                {
                                cout<<"\nPlease ensure a number was entered for the reference.";
                                getch();
```

```
                                uniqueref=0;
                                return uniqueref;
                                }//end else
                        }//end for
        }//end if
        else
                        {
                        //Error message is outputted to alert the user no data has been entered
                        cout<<"\nPlease ensure an  invoice reference has been entered.";
                        getch();
                        uniqueref=0;
                        return uniqueref;
                        }//end else
return uniqueref;
}

//Validation routine that checks the customer entered is on the system
int LocateCustIVal(char custnum[3])
{
//local variables
int compare;
int valid =0;

sscanf(&custnum[0],"%d",& custref);
ifstream fin(FileName2,ios::binary);
fin.seekg(custref*sizeof(a_cust));      //Finds the location of the record in the customer file
fin.get((char*)&a_cust,sizeof(a_cust));
fin.close();

compare=strcmpi(a_cust.flag,"1");      //Checks to see if there is data stored at that location
if(compare==0)
        {                              //If there is, allows the user to enter it
        valid=1;
        }//end if

if(compare!=0)
        {
        //Error message is outputted if there is no data stored at that location in the file
        cout<<"\nCustomer reference not found.";
        }//end if
return valid;
}

//Validation routine to check that the quote reference entered is on the system
int QuoteRefCheck(char quote[3])
{
//local variables
int uniqueref=0;
int find;
int compare;

ReadBackQuotesFile();
for (find = 0; find < nqi; find++)      //searches through all the quotes in the quote file
        {
        compare = strcmpi(quoteref[find], quote);   //Compares each quote reference in the file with the one entered
```

```
        if (compare == 0)  //If cmopare=0 then there is a match between the quote reference entered and one stored in the file
                {
                uniqueref = 1;
                return uniqueref;
                } //endif found record
                else
                        {
                        //Error message is outputted if there is not a match between the quote reference entered and one stored in the file
                        cout<<"\nQuote reference not found.";
                        getch();
                        uniqueref=0;
                        return uniqueref;
                        }//end else
        }//endfor

return uniqueref;
}

//Validation routine that checks the format of the date entered is DD/MM/YYYY
int DateValStart(char date[11])
{
//local variables
int valid =0;
float yearfloat;
int len = strlen(date);    //strlen extracts the number of characetrs entered and stores it under the variable len


if(len == 10)  //Checks that the length is correct
        {
        if(date[2] == '/' && date[5] == '/')   //Checks that the forward slash are in the right place
                {
                sscanf(&date[6],"%d",&yearstart);    //Splits the date entered into days, month and year
                sscanf(&date[3],"%d",&monthstart);
                sscanf(&date[0],"%d",&daystart);
                if(monthstart==4 || monthstart==6 || monthstart==9 || monthstart==11)
                        {
                        if(daystart>= 1 && daystart<= 30)    //If this is the converted month check the days are between 1st and 30th
                                {
                                valid = 1;
                                }//endif-fordays30
                        else
                                {
                                cout<<"Make sure the date is between the 1st and 30th";
                                }//endelse
                        }//endif-formonths30

                if(monthstart==1 || monthstart==3 || monthstart==5 || monthstart==7 || monthstart==8 || monthstart==10 || monthstart==12)
                        {
                        if(daystart>= 1 && daystart<= 31)
                                {                                  //If this is the converted month check the days are between 1st and 31st
                                valid = 1;
                                }//endif-fordays31
                        else
                                {
                                cout<<"Make sure the date is between the 1st and the 31st";
```

```
                        }//endelse
                }//endif-formonths31

        if(monthstart==2)
                {
                yearfloat = yearstart;  //Converts integer to a float for division
                if(floor(yearfloat/4)== (yearfloat/4))      //If month is feb, checks if its a leap year
                        {
                        if(daystart>= 1 && daystart <=29)
                                {
                                //If it is allow up to the 29th
                                valid = 1;
                                }//endif-fordaysfeb29
                        else
                                {
                                cout<<"Make sure the date is between the 1st and the 29th";
                                }//end else
                        }//end if
                        else
                                {
                                if(floor(yearfloat/4)!= (yearfloat/4))      //If its not a leap year only allow up the 28th
                                        {
                                        if(daystart>= 1 && daystart <=28)
                                                {
                                                valid = 1;
                                                }//endif-fordaysfeb28
                                        else
                                                {
                                                cout<<"Make sure the date is between the 1st and the 28th";
                                                }
                                        }//endif
                                }//end else
                        }//endif


                }//endif-forslashes
        else
                {
                cout<<"Forward slashes are in the wrong place";
                }//endelse

        }//endif-forlength
else
        {
        cout<<"Please make sure your date is of length 10";
        }//endelse
return valid;
}

//Validation routine that ensure the date entered is in the format: DD/MM/YYYY
int DateValEnd(char date[11])
{
//local variables
int valid =0;
```

```
int months;
int days;
float years;
int len = strlen(date);

if(len == 10)   //Checks the length is correct
        {
        if(date[2] == '/' && date[5] == '/')    //Checks that the forward slash are in the right place
                {
                sscanf(&date[3],"%d",&months);    //Splits the date entered into days, month and year
                sscanf(&date[0],"%d",&days);
                if(months==4 || months==6 || months==9 || months==11)
                        {
                        if(days>= 1 && days<= 30)
                                {
                                valid = 1;
                                }//endif-fordays30
                        else
                                {
                                cout<<"Make sure the date is between the 1st and 30th";
                                }//endelse
                        }//endif-formonths30

                if(months==1 || months==3 || months==5 || months==7 || months==8 || months==10 || months==12)
                        {
                        if(days>= 1 && days<= 31)
                                {
                                valid = 1;
                                }//endif-fordays31
                        else
                                {
                                cout<<"Make sure the date is between the 1st and the 31st";
                                }//endelse
                        }//endif-formonths31

                if(months==2)
                        {
                        sscanf(&date[7],"%f",&years);    //Converts year to float to allow for division to chek for leap year
                        if(floor(years/4)== (years/4))   //If value of division is the same then leap year
                                {
                                if(days>= 1 && days <=29)
                                        {
                                        valid = 1;
                                        }//endif-fordaysfeb29
                                else
                                        {
                                        cout<<"Make sure the date is between the 1st and the 29th";
                                        }
                                }

                        if(floor(years/4)!= (years/4))   //If value of division is not the same then not a leap year
                                        {
                                        if(days>= 1 && days <=28)
                                                {
                                                valid = 1;
```

```
                                        }//endif-fordaysfeb28
                                else
                                        {
                                        cout<<"Make sure the date is between the 1st and the 28th";
                                        }
                                }//endif
                        }



                }//endif-forslashes
        else
                {
                cout<<"Forward slashes are in the wrong place";
                }//endelse

        }//endif-forlength
else
        {
        cout<<"Please make sure your date is of length 10";
        }//endelse

return valid;
}

//Validation routine that ensure the date entered is after the start date of the job
int RangeDate(char compare[11])
{
//local variables
int day;
int month;
int year;
int valid=0;

sscanf(&compare[6],"%d",&year);
sscanf(&compare[3],"%d",&month);    //extracts values from the date enetered
sscanf(&compare[0],"%d",&day);

if(year>yearstart)
        {            //If year entered is greater than current year then date is in future
        valid =1;
        }
else
        {
        if(year==yearstart)  //Checks that the year is the same as the start date
                {
                if(month>monthstart)
                        {
                        valid =1;       //If year is the same but the month entered is greater than current month then date is in future
                        } //end if
                else
                        {
                        if(month==monthstart)
                                {
```

```
                                if(day>daystart)
                                        {
                                        valid=1;        //If month and year are the sane but the day entered is greater than current day the
                                        }//end if
                                }//end else
                        }//end if
                }//end else


if(valid!=1)
        {
        //Error message is outputted if the date is not after the start date
        cout<<"\nDate entered must be after the start date.";
        }//end if

return valid;
}

//Validation routine to check that the value entered is either 1 or 0
int RangePaid(char pay[2])
{
int valid = 0;
int yescomp;
int nocomp;

yescomp=strcmpi(pay,"Y");  //Compares the payment status entered with Y to see if they match
nocomp=strcmpi(pay,"N");  //Compares the payment status entered with Y to see if they match

if(yescomp==0 || nocomp==0)  //If either of the comparisons above match payment status is accepeted
        {
        valid=1;
        }//end if
        else
                {
                //Error message is outputted if the payment status entered was not Y or N
                cout<<"\nPlease ensure the payment status is Y or N";
                }//end else
return valid;
}

//This function determines whether or not the change invoice menu should be outputted to the screen
int ChangeInvoice()
{
int changestatus=0;

//While loop for outputting change invoice menu
while(changestatus != 3)
        {
        changestatus = ChangeInvoiceMenu();
        }//endwhile
getch();
return 0;
}
```

```
//Routine that allows a user to enter different functions for changing information about an invoice
int ChangeInvoiceMenu()
{
int changeinvoicechoice;
//Outputs change invoice menu to the screen
cout<<"\n \t Change Invoice Information";
cout<<"\n \t *************************\n \n";
cout<<"\n \t 1. Change dates worked";
cout<<"\n \t 2. Change payment status";
cout<<"\n \t 3. Return to Invoice Menu";
cout<<"\n";
cout<<"\n \t Enter choice: ";
getch();
cin>>changeinvoicechoice;   //Users input for choice on menu
clrscr();
switch(changeinvoicechoice) //Allows different functions to be run depending on the menu option entered by the user
        {
        case 1:  //If changeinvoicechoice=1 then the IDates function is ran
             {
             IDates();
             break;
             }
        case 2: //If changeinvoicechoice=2 then the IPay function is ran
             {
             if(level>=2)
                {
                IPay();
                }
             else
                {
                cout<<"\nNot authorised.";
                getch();
                }
             break;
             }

        case 3:  //If changeinvoicechoice=3 then ir returns to the invoice menu
             {
             break;
             }
        default:
               {
               //If user enters a wrong digit following error message is outputted
               cout<<"\nPlease enter a number between 1 and 3.";
               getch();
               break;
               }
        }//endcase


clrscr();
getch();
return changeinvoicechoice;
}
```

```
//Routine to change the dates worked on a job
int IDates()
{
//local variables
char looking[25];
int compare;
int compresult;
int find;
char result[2];
int start=0;
int end=0;
int endrange=0;

char enddate[30];
char startdate[30];
int startbuff=0;
int endbuff=0;

cout<<"\nChange Dates Worked";
cout<<"\n******************";
ReadBackInvoiceFile();
cin.get();

cout<<"\nEnter the invoice reference of the invoice you want to change: ";
cin.getline(looking,25);                                    //Invoice to be found
for(find =0; find<nii; find++)    //Searches through all the invoices in the invoice file
        {
        compare = strcmpi(looking, invoiceref[find]);     //Compares each invoice in the file with the one entered
        if(compare ==0)   //If compare==0, then a match has been found between the reference in the invoice file and the one entered by the
                {
                //Outputs invoice information to the screen
                cout<<"\nCustomer Reference: "<<custnum[find];
                LocateCustInvoice(custnum[find]);             //Routine which finds customer information
                cout<<"\nQuote Reference: "<<quotenum[find];
                LocateQuoteCheck(quotenum[find]);             //Routine which finds quote/booking information
                cin.get();
                cout<<"\nIs this the correct invoice to be amended (Y/N): ";
                cin>>result; //User confirms whether or not is it the correct customer
                compresult=strcmpi(result,"Y"); //Compares input with Y
                if(compresult == 0) //If the input is Y then the new dates can be added, if it is N returns to the Change Invoice Menu
                        {
                        cin.get();
                        //Validates new dates upon entry
                        while(start==0||startbuff==0)      //While loop that controls the validation function which ensures start date is in
                                {
                                cout<<"\nEnter start date: ";
                                cin.getline(startdate,30);
                                startbuff = Buffer(startdate,10);
                                if(startbuff!=0 && start==0)
                                        {
                                        start = DateValStart(jobstartdate[find]);
                                        }//end if
                                }// end while
                        strcpy(jobstartdate[find],startdate);
                        while(end==0 || endrange==0 || endbuff==0)
```

```
                                {
                                cout<<"\nEnter end date: ";  //While loop that controls the validation function which ensures the end date i
                                cin.getline(enddate,30);
                                endbuff = Buffer(enddate,10);
                                if(endbuff!=0 && (endrange==0 || end==0))
                                        {
                                        end = DateValEnd(enddate);
                                        endrange = RangeDate(enddate);
                                        }//endif
                                }// end while
                        strcpy(jobenddate[find],enddate);
                        ReWriteInvoiceFile();  //Wrires information to the invoice file
                        }//end if
                        return 0;
                }// end if
        }//end for
if(compare!=0)
        {
        //If a match has not been found between the reference in the invoice file and the one entered by the user then this error message is
        cout<<"\nInvoice reference not found.";
        }//end if
getch();
return 0;
}

// Allows the user to change the payment status to indicate if the job has been paid for
int IPay()
{
//local variables
char looking[25];
int compare;
int compresult;
int find;
char result[2];
int payment=0;
char payin[30];
int paybuff=0;

cout<<"\nChange Payment Status";
cout<<"\n**********************";
ReadBackInvoiceFile();
cin.get();

cout<<"\nEnter the invoice reference of the invoice you want to change: ";
cin.getline(looking,25);                //Invoice to be found
for(find =0; find<nii; find++)          //Searches through all the invoices in the file
        {
        compare = strcmpi(looking, invoiceref[find]);  //Compares each invoice reference in the file with the reference entered
        if(compare ==0)  //If compare==0, then a match has been found between the reference in the invoice file and the one entered by the u
                {
                //Information regrading the invoice is outputted
                cout<<"\nCustomer Reference: "<<custnum[find];
                LocateCustInvoice(custnum[find]);             //Finds customer information from the customer file
                cout<<"\nQuote Reference: "<<quotenum[find];
                LocateQuoteCheck(quotenum[find]);           //Finds quote/booking information from the quotes file
```

```
                cin.get();
                cout<<"\nIs this the correct invoice to be amended (Y/N): ";
                cin>>result; //User confirms whether or not is it the correct customer
                compresult=strcmpi(result,"Y"); //Compares input with Y
                if(compresult == 0) //If the input is Y then the new dates can be added, if it is N returns to the Change Invoice Menu
                        {
                        cin.get();
                        //Validates payment status upon entry
                        while(payment ==0 || paybuff==0)     //While loop that controls the validation function to ensure the payment status
                                {
                                cout<<"\nIs the job paid for (Y/N): ";
                                cin.getline(payin,30);
                                paybuff = Buffer(payin,1);
                                if(paybuff!=0 && payment==0)
                                        {
                                        payment = RangePaid(payin);
                                        }//endif
                                }// end while
                        strcpy(paid[find],payin);
                        ReWriteInvoiceFile(); //Writes new information to the file
                        }//end if
                        return 0;
                }// end if
        }//end for
if(compare!=0)
        {
        //If a match has not been found between the reference in the invoice file and the one entered by the user then this error message is
        cout<<"\nInvoice reference not found.";
        }//end if
getch();
return 0;
}

// Finds quote information to output with the invoice checks
int LocateQuoteCheck(char quotenum[3])
{
//local variables
int find;
int compare;

ReadBackQuotesFile();
for(find=0;find<nqi;find++)   //searches through the quotes file
        {
        compare = strcmpi(quotenum,quoteref[find]);  //compares each quote reference with the one entered
        if(compare==0) //If compare=0 then there is a match between the quote reference passed and one stored in the file
                {
                //Data is outputted
                cout<<"\nJob Description: "<<mainjobdesc[find];
                }//end if
        }//end for
getch();
return 0;
}

// Allows the user to delete an invoice from the invoices file
```

```
int DeleteInvoice()
{
//local variables
int find;
int compare;
int compresult;
int del;
char looking[25];
char result[2];

cout<<"\nDelete Invoice";
cout<<"\n***************";
ReadBackInvoiceFile();
cin.get();

cout<<"\nEnter the invoice reference of the invoice you wish to delete: ";
cin.getline(looking,25);                        //Invoice to be found
for(find=0;find<nii;find++)                     //Searches through all the invoices in the file
        {
        compare = strcmpi (looking,invoiceref[find]);//Compares each invoice reference in the file with the one entered
        if(compare==0)  //If compare=0 then there is a match between the invoice reference entered and one stored in the file
                {
                //Invoice information is outputted to check it is the correct invoice
                cout<<"\nCustomer Reference: "<<custnum[find];
                LocateCustInvoice(custnum[find]);        //Functions which finds customer information to also output
                cout<<"\nQuote Reference: "<<quotenum[find];
                LocateQuoteCheck(quotenum[find]); //Functions which finds quote/booking information to also output
                cin.get();
                cout<<"\nIs this the correct invoice to be deleted (Y/N): ";
                cin>>result;   //User confirms whether or not is it the correct customer
                compresult=strcmpi(result,"Y"); //Compares the input with Y
                if(compresult == 0)  //If the input is Y then the invoice can be deleted, if it is N returns to the Invoice Menu
                        {
                        //Loops through and deletes all information for that invoice
                        for(del=find;del<nii;del++)
                                {
                                strcpy(invoiceref[del], invoiceref[del+1]);
                                strcpy(custnum[del], custnum[del+1]);
                                strcpy(quotenum[del], quotenum[del+1]);
                                strcpy(invoicedate[del], invoicedate[del+1]);
                                strcpy(jobstartdate[del], jobstartdate[del+1]);
                                strcpy(paid[del], paid[del+1]);
                                nii=nii-1;    //Decreases the number of invoices stored in the file by 1
                                itoa(nii,nic,10);
                                ReWriteInvoiceFile(); //Deletes information from the file
                                }// end for
                                return 0;
                        }//end if
                        return 0;
                }//end if
        }//end for
if(compare!=0)
        {
        //If a match has not been found between the reference in the invoice file and the one entered by the user then this error message is
        cout<<"\nInvoice reference not found.";
```

```
        }//end if

getch();
return 0;
}

//This function determines whether or not the view invoice menu should be outputted to the screen
int ViewInvoice()
{
int viewstatus=0;
//While loop to output view invoice menu
while(viewstatus != 4)
        {
        viewstatus = ViewInvoiceMenu();
        }//endwhile
getch();
return 0;
}

//Outputs view invoice menu options to the user and allows them to choose one of them to run a specific function
int ViewInvoiceMenu()
{
int viewinvoicechoice;
//Outputs view invoice menu
cout<<"\n \t View Invoices";
cout<<"\n \t *************\n \n";
cout<<"\n \t 1. Search invoice by invoice refernece";
cout<<"\n \t 2. Search invoice by customer reference";
cout<<"\n \t 3. Search for unpaid invoices";
cout<<"\n \t 4. Return to Invoice Menu";
cout<<"\n";
cout<<"\n \t Enter choice: ";
getch();
cin>>viewinvoicechoice;   //Users input for their menu choice
clrscr();
switch(viewinvoicechoice)  //Allows different menu options to be ran depending on the menu option entered by the user
        {
        case 1:  //If viewinvoicechoice=1 then the IRef function can be run
             {
             IRef();
             break;
             }

        case 2:   //If viewinvoicechoice=2 then the ICustRef function can be run
             {
             ICustRef();
             break;
             }

        case 3:  //If viewinvoicechoice=3 then the IUnpaid function can be run
             {
             IUnpaid();
             break;
             }
```

```
        case 4: //If viewinvoicechoice=4 then ot returns to the invoice menu
             {
             break;
             }
        default:
               {
               //If user input was out of range following error message is outputted
               cout<<"\nPlease enter a number between 1 and 4.";
               getch();
               break;
               }
        }//endcase


clrscr();
getch();
return viewinvoicechoice;
}

//Allows user to view an invoice by invoice reference
int IRef()
{
//local variables
char looking[25];
int compare;
int find;

cout<<"\nView Invoice by Invoice Reference";
cout<<"\n*********************************";
ReadBackInvoiceFile();
cin.get();

cout<<"\nEnter invoice reference: ";
cin.getline(looking,25);          //Invoice to be found
for(find=0;find<nii;find++)        //Searches through all the invoices in the file
        {
        compare = strcmpi(looking,invoiceref[find]); //Compares each invoice reference in the file with the one entered
        if(compare == 0) //If compare==0 then there is  match between the invoice reference entered and one stored in the file
                {
                //Invoice information is outputted to the screen
                cout<<"\nCustomer reference: "<<custnum[find];
                LocateCust(custnum[find]);   //Finds customer information to output
                cout<<"\n";
                cout<<"\nDate invoice was produced: "<<invoicedate[find];
                cout<<"\n";
                cout<<"\nDates worked: "<<jobstartdate[find]<<"-"<<jobenddate[find];
                cout<<"\n";
                cout<<"\nQuote reference: "<<quotenum[find];
                LocateQuote(quotenum[find]);   //Finds job information to output
                cout<<"\nPayment Status: "<<paid[find];
                }// end if
        }// end for
if(compare!=0)
        {
        //If here is not match between the invoice reference entered and one stored in the file the following error message is outputted
```

```
        cout<<"\nInvoice reference not found.";
        }//end if
getch();
return 0;
}

//Allows the user to view an invoice by customer reference
int ICustRef()
{
//local variables
int compare;
int find;
char looking[25];

cout<<"\nView by customer reference";
cout<<"\n*************************";

ReadBackInvoiceFile();
cout<<"\nEnter customer reference: ";
cin.get();
cin.getline(looking,25);           //Customer reference to be found

for(find=0;find<nii;find++)       //Searches through all the invoices in the file
        {
        compare = strcmpi(looking,custnum[find]); //Compares each customer reference in the file with the one entered
        if(compare==0) //If compare=0 then there is a match between the customer reference entered and one stored in the file
                {
                //Invoice information is outputted to the screen
                cout<<"\nCustomer reference: "<<custnum[find];
                LocateCustInvoice(custnum[find]);    //Finds and outputs customer information
                cout<<"\n";
                cout<<"\nDate invoice was produced: "<<invoicedate[find];
                cout<<"\n";
                cout<<"\nDates worked: "<<jobstartdate[find]<<"-"<<jobenddate[find];
                cout<<"\n";
                cout<<"\nQuote reference: "<<quotenum[find];
                LocateQuote(quotenum[find]);           //Finds and outputs quote/booking information
                cout<<"\nPayment Status: "<<paid[find];
                } // end if
        }// end for
if(compare!=0)
        {
        //If there is not a match between the customer reference entered and one stored in the file the following error message is outputted
        cout<<"\nCustomer reference not found.";
        }//end if
getch();
return 0;
}

// Finds customer information to output with the invoice informatiom
int LocateCustInvoice(char custnum[3])
{
//local variables
int compare;
```

```
sscanf(&custnum[0],"%d",&custref);
ifstream fin(FileName2, ios::binary);
fin.seekg(custref*sizeof(a_cust));          //Calculation to find location of customer in the file
fin.get((char*)&a_cust,sizeof(a_cust));
fin.close();

compare = strcmpi(a_cust.flag,"0");      //Checks to see if there is data stored at that location
if(compare!=0)
        {                                 //If there is customer information is outputted to the screen
        cout<<"\nCustomer Name: "<<a_cust.fnamecust<<" "<<a_cust.lnamecust;
        cout<<"\nCustomer Mobile Number: "<<a_cust.telnocust;
        getch();
        }// end if
getch();
return 0;
}

// Finds quote information to output with the invoice information
int LocateQuote(char quotenum[3])
{
//local variables
int find;
int compare;

ReadBackQuotesFile();
for(find=0;find<nqi;find++) //searches through all the quotes in the file
        {
        compare = strcmpi(quotenum,quoteref[find]); //compares each quote reference with the one entered
        if(compare==0) //If compare=0 there is a match between the quote reference passed and one in the file
                {
                //Quote information is then outputted
                cout<<"\n Job Description: "<<mainjobdesc[find];
                cout<<"\nPrice Breakdown";
                cout<<"\n***************";
                cout<<"\nMaterials: "<<stockcost[find];
                cout<<"\nLabour: "<<labourq[find];
                cout<<"\nMileage: "<<mileage[find];
                cout<<"\nVAT: "<<vat[find];
                cout<<"\nTotal: "<<totalcost[find];
                }//end if
        }//end for
getch();
return 0;
}

//Outputs invoices that have not yet been paid for
int IUnpaid()
{
//local variables
int find;
int compare;

cout<<"\nView Unpaid Invoices";
cout<<"\n********************";
```

```
ReadBackInvoiceFile();
for(find=0;find<nii;find++)      //searches through all the invoices in the file
        {
        compare = strcmpi("N",paid[find]);  //Compares each payment status in the file with N
        if(compare==0)  //If compare=0 then there is a match between the payment status' in the file and N
                {
                //Customer information is outputted
                cout<<"\nCustomer reference: "<<custnum[find];
                LocateCustInvoice(custnum[find]);  //Function to find further customer information to be outputted
                }// end if
        }//  end for
getch();
return 0;
}

//Reads all invoice information from the file
int ReadBackInvoiceFile()
{
int count;
ifstream fin(FileName5,ios::binary);
fin.read((char*)&nic, sizeof(nic));
sscanf(&nic[0], "%d", &nii);
for(count=0;count<nii;count++)
        {
        fin.getline(invoiceref[count],sizeof(invoiceref[count]));
        fin.getline(custnum[count],sizeof(custnum[count]));
        fin.getline(quotenum[count],sizeof(quotenum[count]));
        fin.getline(invoicedate[count],sizeof(invoicedate[count]));
        fin.getline(jobstartdate[count],sizeof(jobstartdate[count]));
        fin.getline(jobenddate[count],sizeof(jobenddate[count]));
        fin.getline(paid[count],sizeof(paid[count]));
        } //end for
fin.close();
return 0;
}

//Writes in new invoiceinformation to the file
int ReWriteInvoiceFile()
{
int count;
ofstream fout(FileName5, ios::binary);
fout.write((char*)&nic,sizeof(nic));
for(count=0;count<nii;count++)
        {
        fout.write((char*)&invoiceref[count],strlen(invoiceref[count]));
        fout.write("\n",1);
        fout.write((char*)&custnum[count],strlen(custnum[count]));
        fout.write("\n",1);
        fout.write((char*)&quotenum[count],strlen(quotenum[count]));
        fout.write("\n",1);
        fout.write((char*)&invoicedate[count],strlen(invoicedate[count]));
        fout.write("\n",1);
        fout.write((char*)&jobstartdate[count],strlen(jobstartdate[count]));
        fout.write("\n",1);
        fout.write((char*)&jobenddate[count],strlen(jobenddate[count]));
```

```
        fout.write("\n",1);
        fout.write((char*)&paid[count],strlen(paid[count]));
        fout.write("\n",1);
        } // end for
fout.close();
return 0;
}

//This function determines whether or not the stock menu should be outputted to the screen
int Stock()
{
int stockstatus=0;
//While loop to output stock menu
while(stockstatus != 6)
        {
        stockstatus = StockMenu();
        }//endwhile
getch();
return 0;
}

//Outputs staff menu options to the user and allows them to choose one of them
int StockMenu()
{
int stockchoice;
//Outputs stock menu to the screen
cout<<"\n \t Stock Menu";
cout<<"\n \t **********\n \n";
cout<<"\n \t 1. Add Stock";
cout<<"\n \t 2. Change stock information";
cout<<"\n \t 3. Delete stock";
cout<<"\n \t 4. View stock";
cout<<"\n \t 5. View low stock";
cout<<"\n \t 6. Return to Main Menu";
cout<<"\n";
cout<<"\n \t Enter choice: ";
getch();
cin>>stockchoice; //User input to access certain part of the menu
clrscr();
switch(stockchoice) //Allows different functions to be ran deoening on the menu option entered by the user
        {
        case 1:  //If stockchoice=1 then the AddStock function is ran
             {
             AddStock();
             break;
             }
        case 2: //If stockchoice=2 then the ChangeStock function is ran
             {
             if(level==3)   //Only allows access to the fuunction if the user logged on has level of access 3
                {
                ChangeStock();
                }
             else
                {
                //If they do not have the correct level of access following error message is outputted
```

```
            cout<<"\nNot authorised.";
            getch();
            }
        break;
        }

    case 3:  //If stockchoice=3 then the DeleteStock function is ran
        {
        DeleteStock();
        break;
        }

    case 4: //If stockchoice=4 then the ViewStock function is ran
        {
        ViewStock();
        break;
        }
    case 5:  //If stockchoice=5 then the ViewLowStock function is ran
        {
        ViewLowStock();
        break;
        }
    case 6:  //If stockchoice=6 then it returns to the main menu
        {
        break;
        }
    default:
          {
          //If user enters a wrong digit following error message is outputted
          cout<<"\nPlease enter a number between 1 and 6.";
          getch();
          break;
          }
    }//endcase

clrscr();
getch();
return stockchoice;
}

//Allows the user to add a new item of stock to the stock file
int AddStock()
{
//local variables
int ref=0;
int col=0;
int ty=0;
int price=0;
int vol=0;
int quant=0;

int refbuff=0;
char refin[30];

ReadBackStockFile();
```

```
cout<<"\nAdd Stock";
cout<<"\n*********";

//Validation routines for information entered
cin.get();
while(ref==0 || refbuff==0)   //While loop that controls validation to ensure that the stock reference entered isn't already in use
        {
        cout<<"\nEnter stock reference: ";
        cin.getline(refin,30);
        refbuff=Buffer(refin,1);
        if(refbuff!=0 && ref==0)
                {
                ref = UniqueStock(refin);
                }//endif
        } // end while
strcpy(stockref[nsti],refin);

while(col == 0)  //While loop that controls validation to ensure that a colour is entered
        {
        cout<<"\nEnter colour: ";
        cin.getline(colour[nsti],15);
        col = PresColour(colour[nsti]);
        }// end while

while(ty == 0)     //While loop that controls validation to ensure that the type of paint entered is valid
        {
        cout<<"\nEnter type of paint (Matt, Gloss, Silk or Exterior): ";
        cin.getline(type[nsti],15);
        ty = PresType(type[nsti]);
        }// end while

while(price == 0)    //While loop that controls validation to ensure that the stock price is within a suitable range
        {
        cout<<"\nEnter stock price: ";
        cin.getline(stockprice[nsti],15);
        price = RangePrice(stockprice[nsti]);
        }// end while

while(vol == 0)    //While loop that controls validation to ensure that the volume is within a suitable range
        {
        cout<<"\nEnter volume: ";
        cin.getline(volume[nsti],4);
        vol = RangeVolume(volume[nsti]);
        }// end while

while(quant == 0)   //While loop that controls validation to ensure that the quantity is within a suitable range
        {
        cout<<"\nEnter quantity: ";
        cin.getline(quantity[nsti],4);
        quant = RangeQuantity(quantity[nsti]);
        } // end while

nsti = nsti+1;    //Increases the number of stock stored in the file by one
itoa(nsti,nstc,10);
ReWriteStockFile();  //Writes information to the file
```

```
getch();
return 0;
}

//Validation routine that checks the stock reference is not already in use
int UniqueStock(char stock[3])
{
//local variables
int uniqueref=1;
int find;
int compare;
int len;
int position;

len=strlen(stock);   //strlen extracts the number of characetrs entered and stores it under the variable len
if(len==0)    //Checks data is entered
        {
        //Error message is outputted if no data has been entered
        cout<<"\nPlease ensure you have entered a stock reference.";
        uniqueref=0;
        getch();
        return uniqueref;
        }
for(position=0;position<len;position++)     //loops through all characters in inout
        {
        if(isdigit(stock[position]))      //Checks to see if it an integer, if it is the rest of the loop runs
                {
                ReadBackStockFile();
                for(find=0;find<nsti;find++)      //searches through all the stock in the file
                        {
                        compare = strcmpi(stockref[find], stock); //compares the stock references in the file with the one entered
                        if (compare == 0)   //if compare=0 then there is a match between the stock reference passed and the stock reference
                                {
                                //If a match is found, user is not allowed to use that reference
                                uniqueref = 0;
                                } //endif
                        }//endfor
                }//end if
                else
                        {
                        cout<<"\nPlease ensure the stock reference is a number.";
                        uniqueref=0;
                        getch();
                        return uniqueref;
                        }//end else
        }//endfor
if (uniqueref==0)
        {
        //Error message is outputted if that reference is in use
        cout<<"\nStock reference cannot be used. Please try again";
        getch() ;
        } // end if error
return uniqueref;
}
```

```
//Validation routine that checks data has been entered
int PresColour(char presence[30])
{
//local variables
int length;
length=strlen(presence);  //strlen extracts the number of characetrs entered and stores it under the variable length

if(length==0)  //Checks to see if the length is 0 as if it is no data has been entered
        {
        //Error message is outputted if length = 0
        cout<<"\nPlease enter a colour.";
        }//endif
return length;
}

//Validation routine that checks data has been entered
int PresType(char presence[30])
{
//local variables
int valid=0;
int matt;
int gloss;
int silk;
int ext;

matt=strcmpi(presence,"Matt");  //Compares the input with the paint type Matt
gloss=strcmpi(presence,"Gloss"); //Compares the input with the paint type Gloss
silk=strcmpi(presence,"Silk"); //Compares the input with the paint type Silk
ext=strcmpi(presence,"Exterior"); //Compares the input with the paint type Exterior

if(matt==0 || gloss==0 || silk==0 || ext==0)  //If the input matches one of the comparisons above then the input is accepeted
        {
        valid=1;
        }//end if
        else
                {
                //Error message is outputted to the screen if the input does not match any paint types
                cout<<"\nPlease ensure you have entered a valid paint type.";
                }//end else
return valid;
}

//Validation routine that checks that the price of the stock entered is reasonable
int RangePrice(char rprice[3])
{
//local variables
int valid = 0;
int intprice;
int len;
int position;
len= strlen(rprice); //strlen extracts the number of characetrs entered and stores it under the variable length



intprice = atoi(rprice);        //converts price to an integer
```

```
//Checks that an integer has been entered
for(position=0;position<len;position++)
        {
        if(isalpha(rprice[position]))
                {
                cout<<"\nPlease ensure you have entered a number for the price of the stock.";
                getch();
                valid=0;
                return valid;
                }//end if
        }//end for
if(len!=0)   //Checks data has been entered
        {
        if(intprice <=100 && intprice >=10)   //Range check
                {
                valid = 1;
                } // end if
        else
                {   //Error message is outputted if its not in the correct range
                cout<<"\nPlease ensure the price entered is within the range 10-100.";
                }//end else
        }//end if
        else
                {
                //Error message is outputted if no data has been entered
                cout<<"\nPlease ensure a price was entered within the range 10-100";
                }//end else

return valid;
}

//Validation routine that checks the volume entered is reasonable
int RangeVolume(char vol[4])
{
//local variables
int valid = 0;
int intvol;
int len;
len=strlen(vol);   //strlen extracts the number of characetrs entered and stores it under the variable len
int position;

//Checks that an integer has been entered
for(position=0;position<len;position++)
        {
        if(isalpha(vol[position]))
                {
                cout<<"\nPlease ensure you have entered a number for the volume.";
                getch();
                valid=0;
                return valid;
                }//end if
        }//end for

intvol = atoi(vol);  //converts volume to an integer
```

```
if(len!=0)
        {
        if(intvol>=5 && intvol <=60) //Range check
                {
                valid = 1;
                } // end if
        else
                {
                //Error message is outputted if the volume is not in the range
                cout<<"\nPlease ensure the volume entered is in the range 5-60.";
                }//end else
        }//end if
        else
                {
                //Error message is outputted if no data has been entered
                cout<<"\nPlease ensure a volume has been entered within the range 5-60.";
                }//end else
return valid;
}

//Validation routine that checks that the quantity entered is reasonable
int RangeQuantity(char quant[4])
{
//local variables
int valid = 0;
int intquant;
int len;
int position;

len = strlen(quant);   //strlen extracts the number of characetrs entered and stores it under the variable len

intquant = atoi(quant);  //converts it to an integer

//Checks that an integer has been entered
for(position=0;position<len;position++)
        {
        if(isalpha(quant[position]))
                {
                cout<<"\nPlease ensure you have entered a number for the quantity.";
                getch();
                valid=0;
                return valid;
                }//end if
        }//end for

if(len!=0)   //Checks data has been entered
        {
        if(intquant >=1 && intquant <=20)  //Range check
                {
                valid = 1;
                } // end if
        else
                {
                //Error message is outputted if the volume is not in the range
```

```
                cout<<"\nPlease ensure the quantity entered is in the range 1-20.";
                } // end else
        }//end if
        else
                {
                //Error message is outputted if no data has been entered
                cout<<"\nPlease ensure a quantity has been entered in the range 1-20.";
                }//end else
return valid;
}

//This function determines whether or not the change stock menu should be outputted to the screen
int ChangeStock()
{
int changestatus=0;
//While loop to output change stock menu
while(changestatus != 3)
        {
        changestatus = ChangeStockMenu();
        }//endwhile
getch();
return 0;
}

//Outputs change staff menu options to the user and allows them to choose one of them
int ChangeStockMenu()
{
int changestockchoice;
//Outputs change stock menu to the screen
cout<<"\n \t Change Stock Information";
cout<<"\n \t **********************\n \n";
cout<<"\n \t 1. Change price of stock";
cout<<"\n \t 2. Change stock quantities";
cout<<"\n \t 3. Return to Stock Menu";
cout<<"\n";
cout<<"\n \t Enter choice: ";
getch();
cin>>changestockchoice;  //Inputted choice from user
clrscr();
switch(changestockchoice) //Allows different functions to be run depending on the menu option entered by the user
        {
        case 1:  //If changestockchoice=1 then the StPrice function is ran
             {
             StPrice();
             break;
             }
        case 2: //If changestockchoice=1 then the StQuantity function is ran
             {
             StQuantity();
             break;
             }

        case 3:  //If changestockchoice=3 then it returns to the stock menu
             {
             break;
```

```
        }
        default:
                {
                //If a wrong digit is entered following error message is outputted
                cout<<"\nPlease enter a number between 1 and 3.";
                getch();
                break;
                }
        }//endcase


clrscr();
getch();
return changestockchoice;
}

//Allows the user to change the price of an item of stock
int StPrice()
{
//local variables
char looking[25];
int compare;
int compresult;
int find;
char result[2];
int price=0;

cout<<"\nChange stock price";
cout<<"\n******************";
ReadBackStockFile();
cin.get();

cin.get();
cout<<"\nEnter the stock reference of the stock you want to change: ";
cin.getline(looking,25);           //stock reference to be found
for(find =0; find<nsti; find++)    //searches through all the stock in the file
        {
        compare = strcmpi(looking, stockref[find]); //compares the stock references in the file with the one entered
        if(compare ==0)  //If compare ==0 then there is a stock reference in the file the same as the one entered by the user
                {
                //Outputs stock information to the screen to confirm it is the right item of stock
                cout<<"\nCurrent quantity: "<<quantity[find];
                cout<<"\nColour: "<<colour[find];
                cout<<"\nType: "<<type[find];
                cout<<"\nStock Price: "<<stockprice[find];
                cin.get();
                cout<<"\nIs this the correct item of stock to be amended (Y/N): ";
                cin>>result; //User confirms whether or not it is the correct item of stock
                compresult=strcmpi(result,"Y"); //Compares the input with Y
                if(compresult == 0) //If the input is Y then the new price can be added, If input is N returns to Change Stock Menu
                        {
                        cin.get();
                        //Validates stock price upon entry
                        while(price == 0)    //While loop that controls the validation to ensure that the price entered is within a suitable
                              {
```

```
                        cout<<"\nEnter stock price: ";
                        cin.getline(stockprice[find],15);
                        price = RangePrice(stockprice[find]);
                        }// end while
                    ReWriteStockFile();     //New price saved to the file
                    return 0;
                    }// end if
                    return 0;
                }//end if
            }//end for
if(compare!=0)
        {
        //If there is not a stock reference in the file the same as the one entered by the user then the following error message is outputte
        cout<<"\nStock reference has not been found.";
        }//end if
getch();
return 0;
}

//Allows the user to change the quantity of an item of stock
int StQuantity()
{
//local variables
char looking[25];
int compare;
int compresult;
int find;
char result[2];
int quant=0;

cout<<"\nChange stock quantity";
cout<<"\n**********************";
ReadBackStockFile();
cin.get();

cin.get();
cout<<"\nEnter the stock reference of the stock items quantity you want to change: ";
cin.getline(looking,25);            //item of stock to be found
for(find =0; find<nsti; find++)     //searches through all the stock in the file
        {
        compare = strcmpi(looking, stockref[find]);//compares all stock references in the file with the one entered
        if(compare ==0) //If compare ==0 then there is a stock reference in the file the same as the one entered by the user
                {
                //Information about the item of stock is outputted to ensure its the correct item
                cout<<"\nCurrent quantity: "<<quantity[find];
                cout<<"\nColour: "<<colour[find];
                cout<<"\nType: "<<type[find];
                cout<<"\nStock Price: "<<stockprice[find];
                cin.get();
                cout<<"\nIs this the correct item of stock to be amended (Y/N): ";
                cin>>result;  //User confirms whether or not it is the correct item of stock
                compresult=strcmpi(result,"Y"); //Compares the input with Y
                if(compresult == 0) //If the input is Y then the new quantity can be added, If input is N returns to Change Stock Menu
                        {
                        cin.get();
```

```
                        //Validates quantity entered
                        while(quant == 0)   //While loop that controls the validation to ensure that the quantity entered is within a suitab
                                {
                                cout<<"\nEnter quantity: ";
                                cin.getline(quantity[find],3);
                                quant = RangeQuantity(quantity[find]);
                                } // end while
                        ReWriteStockFile();   //new quantity is written to the file
                        return 0;
                        }// end if
                        return 0;
                }//end if
            }//end for
if(compare!=0)
        {
        //If there is not a stock reference in the file the same as the one entered by the user then the following error message is outputte
        cout<<"\nStock reference has not been found.";
        }//end if
getch();
return 0;
}

// Allows the user to delete an item of stock from the stock file
int DeleteStock()
{
//local variables
int find;
int compare;
int compresult;
int del;
char looking[25];
char result[2];
int len;
int position;



cout<<"\nDelete stock";
cout<<"\n************";
ReadBackStockFile();
cin.get();

cout<<"\nEnter the stock reference of the item of stock you wish to delete: ";
cin.getline(looking,25);           //item of stock to be found

len=strlen(looking);

if(len!=0)
        {
        for(find=0;find<nsti;find++)       //searches through all the stock in the file
                {
                compare = strcmpi (looking,stockref[find]); //compares each stock reference in the file with the one entered
                if(compare==0)   //If compare ==0 then there is a stock reference in the file the same as the one entered by the user
                        {
                        //Information regarding that item is outputted to ensure its the correct one to be deleted
```

```
                        cout<<"\nColour: "<<colour[find];
                        cout<<"\nType: "<<type[find];
                        cout<<"\nIs this the correct item of stock to be deleted(Y/N): ";
                        cin>>result; //User confirms whether or not it is the correct item of stock
                        compresult=strcmpi(result,"Y"); //Compares the input with Y
                        if(compresult == 0) //If the input is Y then the new quantity can be added, If input is N returns to Stock Menu
                                {
                                 //Loops through and deletes all information for that item of stock
                                for(del=find;del<nsti;del++)
                                        {
                                        strcpy(stockref[del], stockref[del+1]);
                                        strcpy(quantity[del], quantity[del+1]);
                                        strcpy(colour[del], colour[del+1]);
                                        strcpy(volume[del], volume[del+1]);
                                        strcpy(type[del], type[del+1]);
                                        strcpy(stockprice[del], stockprice[del+1]);
                                        nsti=nsti -1;    //Decreases the number of stock stored in the file by one
                                        itoa(nsti,nstc,10);
                                        ReWriteStockFile();  //Deletes information from the file
                                        }// end for
                                }//end if
                                return 0;
                        }//end if
                }//end for
        }//end if
        else
                {
                //Error messag outputted if no reference is entered
                cout<<"\nNo stock reference has been entered.";
                getch();
                return 0;
                }//end else
if(compare!=0)
        {
        //If there is not a stock reference in the file the same as the one entered by the user then the following error message is outputte
        cout<<"\nStock reference has not been found";
        }//end if

getch();
return 0;
}

//This function determines whether or not the view stock menu should be outputted to the screen
int ViewStock()
{
int viewstatus=0;
//While loop to output view stock menu
while(viewstatus != 3)
        {
        viewstatus = ViewStockMenu();
        }//endwhile
getch();
return 0;
}
```

```
//Outputs view stock menu options to the user and allows them to choose one of them to run a specific function
int ViewStockMenu()
{
//Outputs view stock menu
int viewstockchoice;
cout<<"\n \t View Stock Information";
cout<<"\n \t **********************\n \n";
cout<<"\n \t 1. Search stock by stock ID";
cout<<"\n \t 2. Sort stock quantities low to high";
cout<<"\n \t 3. Return to Stock Menu";
cout<<"\n";
cout<<"\n \t Enter choice: ";
getch();
cin>>viewstockchoice;    //Inputted menu choice from user
clrscr();
switch(viewstockchoice)  //Allows different menu options to be ran depending on the menu option entered by the user
        {
        case 1:  //If viewstockchoice=1 then the StID function is ran
             {
             StID();
             break;
             }
        case 2: //If viewstockchoice=2 then the StSortQ function is ran
             {
             StSortQ();
             break;
             }

        case 3: //If viewstockchoice=3 then it returns to the stock menu
             {
             break;
             }
        default:
               {
               //If user inputs a wrong digit the following error message is outputted
               cout<<"\nPlease enter a number between 1 and 3.";
               getch();
               break;
               }
        }//endcase


clrscr();
getch();
return viewstockchoice;
}

// Allows the user to view details on an item of stock by entering the stock reference
int StID()
{
//local variables
char looking[25];
int compare;
int find;
```

```
cout<<"\nView Stock by Stock Reference";
cout<<"\n***************************";
ReadBackStockFile();
cin.get();

cout<<"\nEnter stock reference: ";
cin.getline(looking,25);          //item of stock to be found
for(find=0;find<nsti;find++)      //searches through all the stock in the file
        {
        compare = strcmpi(looking,stockref[find]);  //compares each stock reference in the file with the one entered
        if(compare == 0) //If compare ==0 then there is a stock reference in the file the same as the one entered by the user
                {
                //Stock information is outputted to the screen
                cout<<"\nColour: "<<colour[find];
                cout<<"\nStock Price: "<<stockprice[find];
                cout<<"\nQuantity: "<<quantity[find];
                cout<<"\nType: "<<type[find];
                cout<<"\nVolume: "<<volume[find];
                getch();
                return 0;
                }// end if
        }// end for
if(compare!=0)
        {
        //If there is not a stock reference in the file the same as the one entered by the user then the following error message is outputte
        cout<<"\nStock reference has not been found.";
        getch();
        }//end if
getch();
return 0;
}

// Sorts the stock quantities from low to high
int StSortQ()
{
//local variables
char sref[10][10];
int intquant[10];
char tempquant[4];
int count;


ReadBackStockFile();
for(count=0;count<nsti;count++)
        {
        //Copies values to temporary variables for sort
        strcpy(tempquant,quantity[count]);
        sscanf(&tempquant[0],"%d",&intquant[count]);
        strcpy(sref[count],stockref[count]);
        getch();
        }//end for
BubbleSort(sref,intquant,nsti);
cout<<"\nSorted Stock Quantities: \n";
PrintArray(sref, intquant,nsti);
```

```
getch();
return 0;
}

void BubbleSort(char sref[][10],int intquant[],int n)
{
//local variables
int position;
char temp[3];


//Base Case
if(n==1)
        {
        return;
        }//end if
// one pass of bubble sort. After this pass the largest element is moved to the end
for(position=0;position<n-1;position++)
        {
          if(intquant[position] > intquant[position+1])
                {
                swap(intquant[position], intquant[position+1]);
                strcpy(temp,sref[position]);
                strcpy(sref[position],sref[position+1]);
                strcpy(sref[position+1],temp);
                }//endif
        }//endfor
//Largest element is fixed, recur for remaining array
BubbleSort(sref,intquant,n-1);
}

void PrintArray(char sref[][10],int intquant[],int n)
{
int position;
for(position=0;position<n;position++)
        {
        cout<<"\nStock Reference: "<<sref[position];
        cout<<"\nQuantity: "<<intquant[position];
        StockInformation(sref[position]);
        cout<<"\n";
        }//end for
getch();
}

//Routine which finds stock information to output in the sort
int StockInformation(char ref[3])
{
//local variables
int find;
int compare;

ReadBackStockFile;

for(find=0;find<nsti;find++)  //loops through all the stock stored in the stock file
        {
```

```
        compare = strcmpi(ref,stockref[find]);   //compares the stock reference being sorted to the ones in the file
        if(compare==0) //If compare==0 then there is a match
                {
                //Outputs stock information to be in the sort
                cout<<"\nColour: "<<colour[find];
                cout<<"\nType: "<<type[find];
                }//end if
        }//end for
getch();
return 0;
}
//Routine that shows which items of stock may need re-ordering
int ViewLowStock()
{
//local variables
int find;
int quant;
ReadBackStockFile();
for(find=0;find<nsti;find++)  //searches through all stock in the file
        {
        quant = atoi(quantity[find]);  //converts quantity of stock to an integer
        if(quant<2)
                {
                //If quantity is low, follwoing information is outputted
                cout<<"\nTHE STOCK BELOW IS CURRENTLY LOW IN STOCK";
                cout<<"\n****************************************";
                cout<<"\nStock reference: "<<stockref[find];
                cout<<"\nQuantity: "<<quantity[find];
                cout<<"\n";
                } // end if
        } // end for
getch();
return 0;
}

//Reads all stock information from the file
int ReadBackStockFile()
{
int count;
ifstream fin(FileName3,ios::binary);
fin.read((char*)&nstc, sizeof(nstc));
sscanf(&nstc[0], "%d", &nsti);
for(count=0;count<nsti;count++)
        {
        fin.getline(stockref[count],sizeof(stockref[count]));
        fin.getline(quantity[count],sizeof(quantity[count]));
        fin.getline(colour[count],sizeof(colour[count]));
        fin.getline(volume[count],sizeof(volume[count]));
        fin.getline(type[count],sizeof(type[count]));
        fin.getline(stockprice[count],sizeof(stockprice[count]));
        } // end for
fin.close();
return 0;
}
```

```
//Writes in new stock information to the file
int ReWriteStockFile()
{
int count;
ofstream fout(FileName3, ios::binary);
fout.write((char*)&nstc,sizeof(nstc));
for(count=0;count<nsti;count++)
        {
        fout.write((char*)&stockref[count],strlen(stockref[count]));
        fout.write("\n",1);
        fout.write((char*)&quantity[count],strlen(quantity[count]));
        fout.write("\n",1);
        fout.write((char*)&colour[count],strlen(colour[count]));
        fout.write("\n",1);
        fout.write((char*)&volume[count],strlen(volume[count]));
        fout.write("\n",1);
        fout.write((char*)&type[count],strlen(type[count]));
        fout.write("\n",1);
        fout.write((char*)&stockprice[count],strlen(stockprice[count]));
        fout.write("\n",1);
        } // end for
fout.close();
return 0;
}

//This function determines whether or not the schedule menu should be outputted to the screen
int Schedule()
{
int schedulestatus=0;
//While loop to output schedule menu
while(schedulestatus != 6)
        {
        schedulestatus = ScheduleMenu();
        }//endwhile
getch();
return 0;
}

//Outputs schedule menu options to the user and allows them to choose one of them
int ScheduleMenu()
{
int schedulechoice;
//Outputs schedule menu to the screen
cout<<"\n \t Schedule Menu";
cout<<"\n \t *************\n \n";
cout<<"\n \t 1. Add Booking";
cout<<"\n \t 2. Change booking information";
cout<<"\n \t 3. Delete booking";
cout<<"\n \t 4. View schedule";
cout<<"\n \t 5. Clear all schedule";
cout<<"\n \t 6. Return to Main Menu";
cout<<"\n";
cout<<"\n \t Enter choice: ";
getch();
cin>>schedulechoice;  //Input from the user
```

```
clrscr();
switch(schedulechoice) //Allows different functions to be ran deoening on the menu option entered by the user
        {
        case 1: //If schedulechoice=1 then the AddBooking function is ran
             {
             AddBooking();
             break;
             }
        case 2: //If schedulechoice=2 then the ChangeBooking function is ran
             {
             if(level>=2)  //User logged on is only allowed access if they have the correct level of access of 2 or 3
                {
                ChangeBooking();
                }
             else
                {
                //If the user logged on does not have the correct level of access then the following error message is outputted
                cout<<"\nNot authorised.";
                getch();
                }
             break;
             }

        case 3: //If schedulechoice=3 then the DeleteBooking function is ran
             {
             if(level>=2)
                {
                DeleteBooking();
                }
             else
                {
                cout<<"\nNot authorised.";
                getch();
                }
             break;
             }

        case 4: //If schedulechoice=4 then the ViewBooking function is ran
             {
             ViewSchedule();
             break;
             }
        case 5:  //If schedulechoice=5 then the ClearBooking function is ran
             {
             ClearSchedule( );
             break;
             }

        case 6:  //If schedulechoice=6 then it returns to the main menu
             {
             break;
             }
        default:
               {
               //If user enters a wrong digit following error message is outputted
```

```
                cout<<"\nPlease enter a number between 1 and 5.";
                getch();
                break;
                }
        }//endcase
clrscr();
getch();
return schedulechoice;
}

//Allows the user to add a booking to the schedule
int AddBooking()
{
//local variables
int staffref=0;
int qref=0;
int dateval=0;
int datesyst=0;
int starttime=0;
int endtime=0;
char bookingdate[12];
int endhour;

char ref[3];
char stime[3];
char etime[3];
char datein[30];
int datebuff=0;

ReadBackScheduleFile();
cout<<"\nAdd Booking";
cout<<"\n***********";

//Validates information upon entry

while(staffref==0)   //While loop that controls validation to ensure that the staff reference entered isn't already in use
        {
        cin.get();
        cout<<"\nEnter staff member: ";
        cin.getline(ref,3);
        staff = atoi(ref);
        staffref = StaffRefCheck(staff);
        }//end while

while(qref==0)  //While loop that controls validation routines to ensure that the quote reference entered exists
        {
        cout<<"\nEnter quote reference: ";
        cin.getline(addref,3);
        quoteno = atoi(addref);
        qref = QuoteRefCheckBooking(quoteno);
        } //end while

cin.get();
while(dateval==0 || datesyst==0 || datebuff==0)  //While loop that controls validation to ensure that the date entered is in the correct for
        {
```

```
        cout<<"\nEnter date of booking: ";
        cin.getline(datein,30);
        datebuff = Buffer(datein,10);
        if(datebuff!=0 && (dateval==0 || datesyst==0))
                {
                date=ConvertDate(datein);
                dateval = DateValBooking(datein);
                datesyst = SystemsClockBooking(datein);
                }//end if
        }//end while

while(starttime==0)   //While loop that controls validation to ensure that the start time entered is within the appropriate range
        {
        cout<<"\nEnter the start time for that day (7-19): ";
        cin.getline(stime,3);
        hour = atoi(stime);
        starttime = StartRange(hour);
        }//end while

while(endtime==0)  //While loop that controls validation to ensure that the end time entered is within the appropriate range
        {
        cout<<"\nEnter the finish time for that day (7-19): ";
        cin.getline(etime,3);
        endhour = atoi(etime);
        endtime = EndRange(endhour);
        }//end while

while(endhour>=hour) //While loop that adds the booking to the schedule for the amount of hours entered
        {
        strcpy(booking[staff-1][date-1][hour-7],addref);
        hour = hour+1;
        }//end while
ReWriteScheduleFile();
getch();
return 0;
}

//Validation routine that checks the staff member exists
int StaffRefCheck(int ref)
{
//local variables
int stafffound=0;
int find;
int compare;
char reference[3];
int len;
int position;


itoa(ref, reference,10);//convers staff reference to a character

len =  strlen(reference);  //strlen extracts the number of characetrs entered and stores it under the variable len

ReadBackStaffFile();
if(len!=0)    //Checks data has actally been entered
```

```
	{
	for(position=0;position<len;position++)
		{
		if(isdigit(reference[position]))
			{
			for(find=0;find<nsi;find++)  //Loops through staff file
				{
				compare = strcmpi(reference,staffref[find]); //Compares reference entered with those in the staff file
				if(compare==0) //If compare=0 then the staff reference entered is the same as one stored in the file
					{
					//Staff member exists
					stafffound=1;
					return stafffound;
					}//end if
					else
						{
						cout<<"\nStaff reference not found.";
						}//end else
				}//end for
			}//end if
			else
				{
				cout<<"\nPlease ensure a number was entered for the staff reference.";
				getch();
				stafffound=0;
				return stafffound;
				}//end else
		}//endrfor
	}//end if
	else
		{
		cout<<"\nPlease enter staff reference.";
		getch();
		stafffound=0;
		return stafffound;
		}//end else


return stafffound;
}

//Validation routine that checks the quote reference is in the system
int QuoteRefCheckBooking(int quoter)
{
//local variables
int uniqueref=0;
int find;
int compare;
char quote[3];
int len;
int position;

itoa(quoter,quote,10); //converts quote reference to a character

ReadBackQuotesFile();
```

```
if(len!=0)
        {
        for(position=0;position<len;position++)
                {
                if(isdigit(quote[position]))
                        {
                        for (find = 0; find < nqi; find++)   //searches through all the quotes in the file
                                {
                                compare = strcmpi(quoteref[find], quote); //compares each quote reference in the file with the one entered
                                if (compare == 0) //If compare=0 then the quote reference entered is the same as one stored in the file
                                        {
                                        //Quote reference has been found
                                        uniqueref = 1;
                                        return uniqueref;
                                        } //endif found record
                                        else
                                                {
                                                //If the quote reference entered is not the same as one stored in the file then the followin
                                                cout<<"\n Quote reference not found.";
                                                getch();
                                                uniqueref=0;
                                                return uniqueref;
                                                }//end else
                                }//endfor
                        }//endif
                        else
                                {
                                //Error message outputted if something other than a number is entered
                                cout<<"\nPlease ensure a number is entered for the quote reference.";
                                getch();
                                uniqueref=0;
                                return uniqueref;
                                }//end else
                }//endfor
        }//endif
        else
                {
                //Error message outputted if nothing is entered
                cout<<"\nPlease ensure a reference was entered.";
                getch();
                uniqueref=0;
                return uniqueref;
                }//end else

return uniqueref;
}

//Validation routine that checks the format of the date entered is DD/MM/YYYY
int DateValBooking(char date[12])
{
//local variables
int valid =0;
int months;
int days;
float years;
```

```
int len = strlen(date);

if(len == 10)  //Checks the length of the date is 10
        {
        if(date[2] == '/' && date[5] == '/')    //Checks the forward slashes are in the correct place
                {
                sscanf(&date[3],"%d",&months);
                sscanf(&date[0],"%d",&days);       //Extracts day and month from the date entered
                if(months==4 || months==6 || months==9 || months==11)
                        {
                        if(days>= 1 && days<= 30)   //Checks date is between 1st and 30th for these months
                                {
                                valid = 1;
                                }//endif-fordays30
                        else
                                {
                                cout<<"Make sure the date is between the 1st and 30th";
                                }//endelse
                        }//endif-formonths30

                if(months==1 || months==3 || months==5 || months==7 || months==8 || months==10 || months==12)
                        {
                        if(days>= 1 && days<= 31)
                                {                       //Checks date is between 1st and 31st for these months
                                valid = 1;
                                }//endif-fordays31
                        else
                                {
                                cout<<"Make sure the date is between the 1st and the 31st";
                                }//endelse
                        }//endif-formonths31

                if(months==2)
                        {
                        sscanf(&date[7],"%f",&years);  //Converts year to a float to allow for division with decimals
                        if(floor(years/4)== (years/4))  //If division is equal then its a leap year
                                {
                                if(days>= 1 && days <=29)
                                        {
                                        valid = 1;
                                        }//endif-fordaysfeb29
                                else
                                        {
                                        cout<<"Make sure the date is between the 1st and the 29th";
                                        }
                                }

                        if(floor(years/4)!= (years/4))   //If division is not equal then not a leap year
                                        {
                                        if(days>= 1 && days <=28)
                                                {
                                                valid = 1;
                                                }//endif-fordaysfeb28
                                        else
                                                {
```

```
                                        cout<<"Make sure the date is between the 1st and the 28th";
                                        }
                                }//endif
                        }//end if - months

                }//endif-forslashes
        else
                {
                cout<<"Forward slashes are in the wrong place";
                }//endelse

        }//endif-forlength
else
        {
        cout<<"Please make sure your date is of length 10";
        }//endelse
return valid;
}

//Validation routine that checks the date entered is in the future
int SystemsClockBooking(char date[12])
{
//local variables
int month;
int year;
int day;
int valid=0;

time_t rawtime; //used to output standard British time and date
time(&rawtime);


strcpy(temptime, ctime(&rawtime)); //converts system time to string for manipulation

sscanf(&temptime[20],"%d", &yeart);
sscanf(&temptime[4],"%3s", &monthchar); //extracts 3 characters only
sscanf(&temptime[8],"%d", &dayst);

//Converts month as string into an integer
monthval = strcmpi(monthchar, "Jan");
if(monthval==0)
        {
        monthint = 1;
        }//endif

monthval = strcmpi(monthchar, "Feb");
if(monthval==0)
        {
        monthint = 2;
        }//endif
monthval = strcmpi(monthchar, "Mar");
if(monthval==0)
        {
        monthint = 3;
        }//endif
```

```
monthval = strcmpi(monthchar, "Apr");
if(monthval==0)
        {
        monthint = 4;
        }//endif
monthval = strcmpi(monthchar, "May");
if(monthval==0)
        {
        monthint = 5;
        }//endif
monthval = strcmpi(monthchar, "Jun");
if(monthval==0)
        {
        monthint = 6;
        }//endif
monthval = strcmpi(monthchar, "Jul");
if(monthval==0)
        {
        monthint = 7;
        }//endif
monthval = strcmpi(monthchar, "Aug");
if(monthval==0)
        {
        monthint = 8;
        }//endif
monthval = strcmpi(monthchar, "Sep");
if(monthval==0)
        {
        monthint = 9;
        }//endif
monthval = strcmpi(monthchar, "Oct");
if(monthval==0)
        {
        monthint = 10;
        }//endif
monthval = strcmpi(monthchar, "Nov");
if(monthval==0)
        {
        monthint = 11;
        }//endif
monthval = strcmpi(monthchar, "Dec");
if(monthval==0)
        {
        monthint = 12;
        }//endif

sscanf(&date[6],"%d", &year);
sscanf(&date[3],"%d", &month);   //Extracts day, month and year from the date entered
sscanf(&date[0],"%d", &day);

if(year>yeart)
        {      //If year entered is bigger than current year
        valid =1;
        }
else
```

```
        {
        if(year==yeart)
                {
                if(month>monthint)
                        {               //If year is equal to year entered but month is bigger than current month
                        valid =1;
                        } //end if
                else
                        {
                        if(month==monthint)
                                {               //If year and month are equal to year and month entered but day is bigger than current day
                                if(day>dayst)
                                        {
                                        valid=1;
                                        }//end if
                                else
                                        {
                                        if(day==dayst)   //If year and month are equal to year and month entered but day is also equal to cu
                                                {
                                                valid=1;
                                                }//end if
                                        }//end else

                                }//end if
                        }//end else
                }//end if
        }//end else
if(valid!=1)
        {
        //If date entered is before the date in the systems clock then the following error message is outputted
        cout<<"\nDate entered is not in the future.";
        }//end if
return valid;
}

//Validation routine that checks the start time for the day is within opening hours
int StartRange(int stime)
{
//local variables
int valid=0;
int position;
int len;
char time[3];

itoa(stime,time,10);

len=strlen(time); //Extracts the number of characters stored

if(len!=0) //Checks data has been entered
        {
        //Loops through all the characters entered
        for(position=0;position<len;position++)
                {
                //Checks input is a number
                if(isdigit(time[position]))
```

```
                        {
                        //Range check
                        if(stime>=7 && stime<=19)
                                {
                                valid=1;
                                }//end if
                                else
                                        {
                                        //If time entered is outside of the range then the following error message is outputted
                                        cout<<"\n Enter a start time within the hours shown above.";
                                        getch();
                                        valid=0;
                                        return valid;
                                        }//end else
                        }//endif
                        else
                                {
                                //Error message is outputted is something other than a number is inputted
                                cout<<"\nPlease ensure the time entered is a number.";
                                getch();
                                valid=0;
                                return valid;
                                }//end else
                }//end for
        }//end if
        else
                {
                //Error message outputted if no data has been entered
                cout<<"\nPlease ensure a start time is entered.";
                getch();
                valid=0;
                return valid;
                }//end else


return valid;
}

//Validation routine that checks the start time for the day is within opening hours
int EndRange(int etime)
{
//local variables
int valid=0;

//Range check
if(etime>=7 && etime<=19)
        {
        valid=1;
        }//end if
if(valid!=1)
        {
        cout<<"\n Enter a finish time within the hours shown above.";
        }//end if
return valid;
}
```

```
// Allows the user to change the date of the booking - keeping hours at work the same
int ChangeBooking()
{
//local variables
int quotenum;
char ref[3];
int newdate;
char bookingdate[12];
char newbookingdate[12];
int endhour;

int staffref=0;
int datevalold=0;
int datevalnew=0;
int starttime=0;
int endtime=0;
int qref=0;

char sref[3];
char stime[3];
char etime[3];

char datein[30];
int datebuff=0;

char newdatein[30];
int newdatebuff=0;

cout<<"\nChange date of booking";
cout<<"\n**********************";
ReadBackScheduleFile();

//Validation upon entry for inputs
while(staffref==0)    //While loop that controls validation to ensure that the staff reference entered exsits
        {
        cin.get();
        cout<<"\nEnter staff member: ";
        cin.getline(sref,3);
        staff = atoi(sref);
        staffref = StaffRefCheck(staff);
        }//end while

while(datevalold==0 || datebuff==0)  //While loop that controls validation to ensure that the date entered is in the correct format: DD/MM/Y
        {
        cout<<"\n Enter old date of booking: ";
        cin.getline(datein,30);
        datebuff = Buffer(datein,10);
        if(datebuff!=0 && datevalold==0)
                {
                date=ConvertDate(bookingdate); //Converts date entered into a number 1-366
                datevalold = DateValBooking(datein);
                }//endif
        }//end while
```

```
while(starttime==0) //While loop that controls validation to ensure that the start time entered is within the appropriate range
        {
        cout<<"\n Enter old start time of booking(7-19): ";
        cin.getline(stime,3);
        hour = atoi(stime);
        starttime = StartRange(hour);
        }//end while

while(endtime==0) //While loop that controls validation to ensure that the end time entered is within the appropriate range
        {
        cout<<"\n Enter old finishing time of booking(7-19): ";
        cin.getline(etime,3);
        endhour = atoi(etime);
        endtime = EndRange(endhour);
        }//end while

while(datevalnew==0 || newdatebuff==0) //While loop that controls validation to ensure that the date entered is in the correct format: DD/MM
        {
        cout<<"\n Enter new date of booking: ";
        cin.getline(newdatein,30);
        newdatebuff = Buffer(newdatein,10);
        if(newdatebuff!=0 && datevalnew==0)
                {
                newdate=ConvertDateNew(newdatein); //Converts date entered into a number 1-366
                datevalnew = DateValBooking(newdatein);
                }//endif
        }//end while

while(qref==0) //While loop that controls validation routines to ensure that the quote reference entered exists
        {
        cout<<"\nEnter quote reference: ";
        cin.getline(ref,3);
        quotenum=atoi(ref);
        qref = QuoteRefCheckBooking(quotenum);
        }

//Deletes old booking from the schedule
while(endhour>=hour) //While loop that deletes the booking to the schedule for the amount of hours entered
        {
        strcpy(booking[staff-1][date-1][hour-7],"*");
        hour = hour+1;
        }//end while

//Adds amended booking to the schedule
date = newdate;
hour=atoi(stime);
while(endhour>=hour)  //While loop that adds the booking to the schedule for the amount of hours entered
        {
        strcpy(booking[staff-1][date-1][hour-7],ref);
        hour=hour+1;
        }//end while
ReWriteScheduleFile();
getch();
return 0;
}
```

```
// Allows the user to delete a booking from the schedule
int DeleteBooking()
{
//local variables
char bookingdate[12];
int endhour;
int staffref=0;
int dateval=0;
int starttime=0;
int endtime=0;

char sref[3];
char stime[3];
char etime[3];

int datebuff=0;
char datein[30];
cout<<"\nDelete Booking";
cout<<"\n**************";
ReadBackScheduleFile();

//Validates inputs upon entry
while(staffref==0)   //While loop that controls validation to ensure that the staff reference entered exsits
        {
        cin.get();
        cout<<"\nEnter staff reference: ";
        cin.getline(sref,3);
        staff = atoi(sref);
        staffref = StaffRefCheck(staff);
        }//end while

while(dateval==0 || datebuff==0)  //While loop that controls validation to ensure that the date entered is in the correct format: DD/MM/YYYY
        {
        cout<<"\nEnter the date that the booking has been cancelled: ";
        cin.getline(datein,30);
        datebuff = Buffer(datein,10);
        if(datebuff!=0&&dateval==0)
                {
                date=ConvertDate(datein); //Converts date entered into a number 1-366
                dateval = DateValBooking(datein);
                }//endif
        }//end while

while(starttime==0) //While loop that controls validation to ensure that the start time entered is within the appropriate range
        {
        cout<<"\nEnter the start time for the cancelled day (7-19): ";
        cin.getline(stime,3);
        hour = atoi(stime);
        starttime = StartRange(hour);
        }//end while

while(endtime==0) //While loop that controls validation to ensure that the end time entered is within the appropriate range
        {
        cout<<"\nEnter the finish time for the cancelled day (7-19): ";
```

```
        cin.getline(etime,3);
        endhour = atoi(etime);
        endtime = EndRange(endhour);
        }//end while

while(endhour>=hour) //While loop that deletes the booking to the schedule for the amount of hours entered
        {
        strcpy(booking[staff-1][date-1][hour-7],"*");
        hour = hour+1;
        }
ReWriteScheduleFile();
getch();
return 0;
}

// Allows the user to remove all booking from all staff schedules
int ClearSchedule()
{
for(staff=0;staff<2;staff++)
        {
        for(date=0;date<366;date++)
                {
                for(hour=0;hour<13;hour++)
                        {
                        strcpy(booking[staff][date][hour],"*");
                        }//endfor - hour
                }//endfor - date
        }//endfor - staff
ReWriteScheduleFile();
return 0;
}

//This function determines whether or not the view schedule menu should be outputted to the screen
int ViewSchedule()
{
int viewstatus=0;
//While loop to output view schedule menu
while(viewstatus != 4)
        {
        viewstatus = ViewScheduleMenu();
        }//endwhile
getch();
return 0;
}

//Outputs view schedule menu options to the user and allows them to choose one of them to run a specific function
int ViewScheduleMenu()
{
int viewschedulechoice;
//Outputs view schedule menu
cout<<"\n \t View Schedules";
cout<<"\n \t **************\n \n";
cout<<"\n \t 1. View Staff 1";
cout<<"\n \t 2. View Staff 2";
cout<<"\n \t 3. View Todays Work";
```

```
cout<<"\n \t 4. Return to Schedule Menu";
cout<<"\n";
cout<<"\n \t Enter choice: ";
getch();
cin>>viewschedulechoice;  //user input from menu choices
clrscr();
switch(viewschedulechoice) //Allows different menu options to be ran depending on the menu option entered by the user
        {
        case 1: //If viewschedulechoice=1 then the Staff1 function is ran
             {
             Staff1();
             break;
             }
        case 2:  //If viewschedulechoice=2 then the Staff2 function is ran
             {
             Staff2();
             break;
             }

        case 3: //If viewschedulechoice=1 then the TodaysWork function is ran
             {
             TodaysWork();
             break;
             }
        case 4: //If viewschedulechoice=1 then it returns to the schedule menu
             {
             break;
             }
        default:
               {
               //If user enters a wrong digit following error message is outputted
               cout<<"\nPlease enter a number between 1 and 4.";
               getch();
               break;
               }
        }//endcase


clrscr();
getch();
return viewschedulechoice;
}

// Allows the user to view the schedule of staff member 1
int Staff1()
{
ReadBackScheduleFile();
cout<<"\nView Booking";
cout<<"\n************";
for(staff=0;staff<1;staff++) //Loops for staff member 1
        {
        cout<<"\nStaff Reference: "<<staff +1;
        cout<<"\n******************";
        cout<<"\n\t\t07\t08\t09\t10\t11\t12\t13\t14\t15\t16\t17\t18\t19";  //Outputs working hours
        for(date=0;date<366;date++)
```

```
                {
                DateOutput(date); //Function to output dates
                for(hour=0;hour<13;hour++)
                        {
                        cout<<"\t" <<booking[staff][date][hour];  //Outputs * for no booking or quote reference to the screen
                        }//endfor - hour
                }//endfor - date
        }// endfor - staff
getch();
return 0;
}

// Allows the user to view the schedule of staff member 2
int Staff2()
{
ReadBackScheduleFile();
cout<<"\nView Booking";
cout<<"\n************";
for(staff=1;staff<2;staff++) //Loops for staff member 2
        {
        cout<<"\nStaff Reference: "<<staff +1;
        cout<<"\n*****************";
        cout<<"\n\t\t07\t08\t09\t10\t11\t12\t13\t14\t15\t16\t17\t18\t19"; //Outputs working hours
        for(date=0;date<366;date++)
                {
                DateOutput(date); //Function to output dates
                for(hour=0;hour<13;hour++)
                        {
                        cout<<"\t" <<booking[staff][date][hour];   //Outputs * for no booking or quote reference to the screen
                        }//endfor - hour
                }//endfor - date
        }// endfor - staff
getch();
return 0;
}

//Ouputs the dates from 01/01/2024 to 31/12/2024
int DateOutput(int dateofyear)
{
//local variables
int leap;

time_t rawtime; //used to output standard British time and date
time(&rawtime);
strcpy(temptime, ctime(&rawtime)); //converts system time to string for manipulation
sscanf(&temptime[20],"%d", &yeart);

dateofyear = dateofyear+1;

if((floor(yeart/4))== ((yeart/4)))    //checks for leap year
        {
        leap=1;
        }
else
        {
```

```
        leap=0;
        }

if(leap==1)
        {
        //cout<<"\n";
        if(dateofyear<32)  //JANUARY
                {
                cout<<"\n"<<dateofyear<<"/"<<"01"<<"/"<<yeart;
                }
        if(dateofyear>31 && dateofyear<61)    //FEBRUARY
                {
                cout<<"\n"<<dateofyear-31<<"/"<<"02"<<"/"<<yeart;
                }
        if(dateofyear>60 && dateofyear<92)   //MARCH
                {
                cout<<"\n"<<dateofyear-60<<"/"<<"03"<<"/"<<yeart;
                }
        if(dateofyear>91 && dateofyear<122)   //APRIL
                {
                cout<<"\n"<<dateofyear-91<<"/"<<"04"<<"/"<<yeart;
                }
        if(dateofyear>121 && dateofyear<153)   //MAY
                {
                cout<<"\n"<<dateofyear-121<<"/"<<"05"<<"/"<<yeart;
                }
        if(dateofyear>152 && dateofyear<183)   //JUNE
                {
                cout<<"\n"<<dateofyear-152<<"/"<<"06"<<"/"<<yeart;
                }
        if(dateofyear>182 && dateofyear<214)  //JULY
                {
                cout<<"\n"<<dateofyear-182<<"/"<<"07"<<"/"<<yeart;
                }
        if(dateofyear>213 && dateofyear<245)   //AUGUST
                {
                cout<<"\n"<<dateofyear-213<<"/"<<"08"<<"/"<<yeart;
                }
        if(dateofyear>244 && dateofyear<275)  //SEPTEMBER
                {
                cout<<"\n"<<dateofyear-244<<"/"<<"09"<<"/"<<yeart;
                }
        if(dateofyear>274 && dateofyear<306)  //OCTOBER
                {
                cout<<"\n"<<dateofyear-274<<"/"<<"10"<<"/"<<yeart;
                }
        if(dateofyear>305 && dateofyear<336) //NOVEMBER
                {
                cout<<"\n"<<dateofyear-305<<"/"<<"11"<<"/"<<yeart;
                }
        if(dateofyear>335 && dateofyear<367) //DECEBER
                {
                cout<<"\n"<<dateofyear-335<<"/"<<"12"<<"/"<<yeart;
                }
```

```
        }//end if - leap

if(leap==0)
        {
        if(dateofyear<32)  //JANUARY
                {
                cout<<"\n"<<dateofyear<<"/"<<"01"<<"/"<<yeart;
                }
        if(dateofyear>31 && dateofyear<60)    //FEBRUARY
                {
                cout<<"\n"<<dateofyear-31<<"/"<<"02"<<"/"<<yeart;
                }
        if(dateofyear>59 && dateofyear<91)   //MARCH
                {
                cout<<"\n"<<dateofyear-59<<"/"<<"03"<<"/"<<yeart;
                }
        if(dateofyear>90 && dateofyear<121)   //APRIL
                {
                cout<<"\n"<<dateofyear-90<<"/"<<"04"<<"/"<<yeart;
                }
        if(dateofyear>120 && dateofyear<152)   //MAY
                {
                cout<<"\n"<<dateofyear-120<<"/"<<"05"<<"/"<<yeart;
                }
        if(dateofyear>151 && dateofyear<182)   //JUNE
                {
                cout<<"\n"<<dateofyear-151<<"/"<<"06"<<"/"<<yeart;
                }
        if(dateofyear>181 && dateofyear<213)  //JULY
                {
                cout<<"\n"<<dateofyear-181<<"/"<<"07"<<"/"<<yeart;
                }
        if(dateofyear>212 && dateofyear<244)   //AUGUST
                {
                cout<<"\n"<<dateofyear-212<<"/"<<"08"<<"/"<<yeart;
                }
        if(dateofyear>243 && dateofyear<274)  //SEPTEMBER
                {
                cout<<"\n"<<dateofyear-243<<"/"<<"09"<<"/"<<yeart;
                }
        if(dateofyear>273 && dateofyear<305)  //OCTOBER
                {
                cout<<"\n"<<dateofyear-273<<"/"<<"10"<<"/"<<yeart;
                }
        if(dateofyear>304 && dateofyear<335) //NOVEMBER
                {
                cout<<"\n"<<dateofyear-304<<"/"<<"11"<<"/"<<yeart;
                }
        if(dateofyear>334 && dateofyear<366) //DECEBER
                {
                cout<<"\n"<<dateofyear-334<<"/"<<"12"<<"/"<<yeart;
                }

        }//end if
return dateofyear;
```

```
}

//Shows what is booked on the schedule for that day
int TodaysWork()
{
//local variables
int leap;
int value;
int date;
char tempquote[3];

time_t rawtime; //used to output standard British time and date
time(&rawtime);
strcpy(temptime, ctime(&rawtime)); //converts system time to string for manipulation
sscanf(&temptime[20],"%d", &yeart);
sscanf(&temptime[4],"%s3", &monthchar);
sscanf(&temptime[8],"%d", &dayst);


cout<<"\n View Todays Work: "<<dayst<<" "<<monthchar<<" "<<yeart;
cout<<"\n ***************************";

if((floor(yeart/4))== ((yeart/4)))    //checks for leap year
        {
        leap=1;
        }
else
        {
        leap=0;
        }

//Converts month entered as a string to an integer
monthval = strcmpi(monthchar, "Jan");
if(monthval==0)
        {
        monthint = 1;
        }//endif

monthval = strcmpi(monthchar, "Feb");
if(monthval==0)
        {
        monthint = 2;
        }//endif
monthval = strcmpi(monthchar, "Mar");
if(monthval==0)
        {
        monthint = 3;
        }//endif
monthval = strcmpi(monthchar, "Apr");
if(monthval==0)
        {
        monthint = 4;
        }//endif
monthval = strcmpi(monthchar, "May");
if(monthval==0)
```

```
        {
        monthint = 5;
        }//endif
monthval = strcmpi(monthchar, "Jun");
if(monthval==0)
        {
        monthint = 6;
        }//endif
monthval = strcmpi(monthchar, "Jul");
if(monthval==0)
        {
        monthint = 7;
        }//endif
monthval = strcmpi(monthchar, "Aug");
if(monthval==0)
        {
        monthint = 8;
        }//endif
monthval = strcmpi(monthchar, "Sep");
if(monthval==0)
        {
        monthint = 9;
        }//endif
monthval = strcmpi(monthchar, "Oct");
if(monthval==0)
        {
        monthint = 10;
        }//endif
monthval = strcmpi(monthchar, "Nov");
if(monthval==0)
        {
        monthint = 11;
        }//endif
monthval = strcmpi(monthchar, "Dec");
if(monthval==0)
        {
        monthint = 12;
        }//endif

//Converts date to a number 1-366
if(monthint==1)
        {
        value = dayst;
        }
if(monthint==2)
        {
        value = dayst +31;
        }

if(monthint>=3 && leap==1)
        {
        if(monthint==3)
                {
                value = dayst + 31 + 29;
                }
```

```
        if(monthint==4)
                {
                value = dayst + 31+ 29 + 31;
                }
                if(monthint==5)
                        {
                        value = dayst + 31+ 29 + 31 +30;
                        }
                        if(monthint==6)
                                {
                                value = dayst + 31+ 29 + 31 +30 + 31;
                                }
                                if(monthint==7)
                                        {
                                        monthint = dayst + 31+ 29 + 31+30+31+30;
                                        }
                                        if(monthint==8)
                                                {
                                                value = dayst + 31+ 29 + 31+30+31+30+31;
                                                }
                                                if(monthint==9)
                                                        {
                                                        value = dayst + 31+29+31+30+31+30+31+31;
                                                        }
                                                        if(monthint==10)
                                                                {
                                                                value = dayst + 31+29+31+30+31+30+31+31+30;
                                                                }
                                                                if(monthint==11)
                                                                        {
                                                                        value = dayst + 31+29+31+30+31+30+31+31+30+31;
                                                                        }
                                                                        if(monthint==12)
                                                                                {
                                                                                value = dayst + 31+29+31+30+31+30+31+31+30+31+30;
                                                                                }
         }//end if -leap year


else
        {
        if(monthint==3)
                {
                value = dayst + 31 + 28;
                }
        if(monthint==4)
                {
                value = dayst + 31+ 28 + 31;
                }
        if(monthint==5)
                {
                value = dayst + 31+ 28 + 31 +30;
                }
        if(monthint==6)
```

```
                {
                value = dayst + 31+ 28 + 31 +30 + 31;
                }

        if(monthint==7)
                {
                value = dayst + 31+ 28 + 31+30+31+30;
                }
        if(monthint==8)
                {
                value = dayst + 31+ 28 + 31+30+31+30+31;
                }
        if(monthint==9)
                {
                value = dayst + 31+28+31+30+31+30+31+31;
                }
        if(monthint==10)
                {
                value = dayst + 31+28+31+30+31+30+31+31+30;
                }
        if(monthint==11)
                {
                value = dayst + 31+28+31+30+31+30+31+31+30+31;
                }
        if(monthint==12)
                {
                value = dayst + 31+28+31+30+31+30+31+31+30+31+30;
                }
        }
date=value-1;
ReadBackScheduleFile();
for(staff=0;staff<2;staff++) //Loops through the staff reference
        {
        cout<<"\nStaff Reference: "<<staff +1;
        cout<<"\n******************";
        cout<<"\n";
        for(hour=0;hour<13;hour++)
                {
                cout<<"\n"<<hour+7<<"\t";  //Outputs working houts
                cout<<booking[staff][date][hour];
                if(strcmpi(tempquote,booking[staff][date][hour])!=0)  //If the quote reference is not the same as the temportary quote infor
                        {
                        LocateQuoteWork(booking[staff][date][hour]);  //Finds quote information to output
                        }//end if
                        strcpy(tempquote,booking[staff][date][hour]);  //Copies the quote reference to the temporary quote
                }//endfor - hour
        }// endfor - staff

getch();
return 0;
}

// Finds quote information to output with the today's work routine
int LocateQuoteWork(char charref[3])
{
```

```
//local variables
int find;
int compare;

ReadBackQuotesFile();
for(find=0;find<nqi;find++) //searches through all the quotes in the file
        {
        compare = strcmpi(charref,quoteref[find]); //compares each quote reference with the one entered
        if(compare==0)   //If compare=0 then the quote reference has been found in the file
                {
                //Outputs quote information to the screen
                cout<<"\n Job Information";
                cout<<"\n*****************";
                cout<<"\nQuote reference: "<<quoteref[find];
                cout<<"\nJob Description: "<<mainjobdesc[find];
                cout<<"\n\n\t";
                cout<<"\nCustomer Information";
                cout<<"\n*********************";
                cout<<"\n\t";
                cout<<"\nCustomer reference: "<<custno[find];
                LocateCustSchedule(custno[find]);                       //If a match is found, information is outputted to the screen
                }//end if
        }//end for
getch();
return 0;
}

//Uses the customer reference to link customer information to output for today's work routine
int LocateCustSchedule(char custno[3])
{
int compare;

sscanf(&custno[0],"%d",& custref);        //Calculation to locate the customer in the file
ifstream fin(FileName2,ios::binary);
fin.seekg(custref*sizeof(a_cust));
fin.get((char*)&a_cust,sizeof(a_cust));
fin.close();

compare=strcmpi(a_cust.flag,"0");        //Checks to see if the location in the file is empty
if(compare!=0)     //If compare!=0 then the location is occupied
        {
        //Customer information is outputted
        cout<<"\nCustomer Name: "<<a_cust.title<<" "<< a_cust.fnamecust<<" "<<a_cust.lnamecust;
        cout<<"\nCustomer Mobile Number: "<<a_cust.telnocust;
        cout<"\n\n";
        cout<<"\nHome Address:"<<a_cust.oneadcust<<"\n             "<<a_cust.twoadcust<<"\n             "<<a_cust.threeadcust<<"\n
        getch();
        }//endif

getch();
return 0;
}

//Reads all schedule from the file
int ReadBackScheduleFile()
```

```
{
ifstream fin(FileName7, ios::in);
for(staff=0;staff<2;staff++)
       {
        for(date=0;date<366;date++)
                {
                for(hour=0;hour<13;hour++)
                        {
                        fin.getline((char*)&booking[staff][date][hour],3);
                        }//end for hour
                }//endfor date
        }//endfor - staff
getch();
fin.close();
return 0;
}

//Writes in new schedule information to the file
int ReWriteScheduleFile()
{
ofstream fout(FileName7, ios::binary);
for(staff=0;staff<2;staff++)
        {
        for(date=0;date<366;date++)
                {
                for(hour=0;hour<13;hour++)
                        {
                        fout.write((char*)&booking[staff][date][hour],1);
                        fout.write("\n",1);
                        }//endfor - hour
                }//endfor - date
        }//  endfor - staff
        getch();
fout.close();
return 0;
}

//Reads all links information from the file
int ReadBackLinksFile()
{
int count;
ifstream fin(FileName6,ios::binary);
fin.read((char*)&nlc, sizeof(nlc));
sscanf(&nlc[0], "%d", &nli);
for(count=0;count<nli;count++)
        {
        fin.getline(linksquoteref[count],sizeof(linksquoteref[count]));
        fin.getline(linksstockref[count],sizeof(linksstockref[count]));
        fin.getline(linksquantity[count],sizeof(linksquantity[count]));
        } // end for
fin.close();
return 0;
}

//Writes in new links information to the file
```

```
int ReWriteLinksFile()
{
int count;
ofstream fout(FileName6, ios::binary);
fout.write((char*)&nlc,sizeof(nlc));
for(count=0;count<nli;count++)
        {
        fout.write((char*)&linksquoteref[count],strlen(linksquoteref[count]));
        fout.write("\n",1);
        fout.write((char*)&linksstockref[count],strlen(linksstockref[count]));
        fout.write("\n",1);
        fout.write((char*)&linksquantity[count],strlen(linksquantity[count]));
        fout.write("\n",1);
        } //end for
fout.close();
return 0;
}


//Converts date in format DD/MM/YYYY to the day of the year for the for loop
ConvertDate(char datein[12])
{
//local variables
float year;
int month;
int day;
int value=0;
int valid=0;
sscanf(&datein[6],"%f", &year);
sscanf(&datein[4],"%d", &month);  //Extracts day, month and year from date entered
sscanf(&datein[0],"%d", &day);

if((floor(year/4))== ((year/4)))    //checks for leap year
        {
        valid=1;
        }
else
        {
        valid=0;
        }

if(month==1)
        {
        value = day;
        }
if(month==2)
        {
        value = day +31;
        }

//Converts date entered into a number 1-366
if(month>=3 && valid==1)
        {
        if(month==3)
                {
```

```
        value = day + 31 + 29;
        }
        if(month==4)
                {
                value = day + 31+ 29 + 31;
                }
                if(month==5)
                        {
                        value = day + 31+ 29 + 31 +30;
                        }
                        if(month==6)
                                {
                                value = day + 31+ 29 + 31 +30 + 31;
                                }
                                if(month==7)
                                        {
                                        value = day + 31+ 29 + 31+30+31+30;
                                        }
                                        if(month==8)
                                                {
                                                value = day + 31+ 29 + 31+30+31+30+31;
                                                }
                                                if(month==9)
                                                        {
                                                        value = day + 31+29+31+30+31+30+31+31;
                                                        }
                                                        if(month==10)
                                                                {
                                                                value = day + 31+29+31+30+31+30+31+31+30;
                                                                }
                                                                if(month==11)
                                                                        {
                                                                        value = day + 31+29+31+30+31+30+31+31+30+31;
                                                                        }
                                                                        if(month==12)
                                                                                {
                                                                                value = day + 31+29+31+30+31+30+31+31+30+31+30;
                                                                                }
 }


else
        {
        if(month==3)
                {
                value = day + 31 + 28;
                }
        if(month==4)
                {
                value = day + 31+ 28 + 31;
                }
        if(month==5)
                {
                value = day + 31+ 28 + 31 +30;
```

```
                }
        if(month==6)
                {
                value = day + 31+ 28 + 31 +30 + 31;
                }

        if(month==7)
                {
                value = day + 31+ 28 + 31+30+31+30;
                }
        if(month==8)
                {
                value = day + 31+ 28 + 31+30+31+30+31;
                }
        if(month==9)
                {
                value = day + 31+28+31+30+31+30+31+31;
                }
        if(month==10)
                {
                value = day + 31+28+31+30+31+30+31+31+30;
                }
        if(month==11)
                {
                value = day + 31+28+31+30+31+30+31+31+30+31;
                }
        if(month==12)
                {
                value = day + 31+28+31+30+31+30+31+31+30+31+30;
                }
        }
return value;
}

//Converts date in format DD/MM/YYYY to the day of the year for the for loop
ConvertDateNew(char newdatein[12])
{
//local variables
float year;
int month;
int day;
int value=0;
int valid=0;
sscanf(&newdatein[6],"%f", &year);
sscanf(&newdatein[4],"%d", &month);   //Extracts day, month and year from date entered
sscanf(&newdatein[0],"%d", &day);

if((floor(year/4))== ((year/4)))        //checks for leap year
        {
        valid=1;
        }
else
        {
        valid=0;
        }
```

```
if(month==1)
        {
        value = day;
        }
if(month==2)
        {
        value = day +31;
        }

if(month>=3 && valid==1)
        {
        if(month==3)
                {
                value = day + 31 + 29;
                }
                if(month==4)
                        {
                        value = day + 31+ 29 + 31;
                        }
                        if(month==5)
                                {
                                value = day + 31+ 29 + 31 +30;
                                }
                                if(month==6)
                                        {
                                        value = day + 31+ 29 + 31 +30 + 31;
                                        }
                                        if(month==7)
                                                {
                                                value = day + 31+ 29 + 31+30+31+30;
                                                }
                                                if(month==8)
                                                        {
                                                        value = day + 31+ 29 + 31+30+31+30+31;
                                                        }
                                                        if(month==9)
                                                                {
                                                                value = day + 31+29+31+30+31+30+31+31;
                                                                }
                                                                if(month==10)
                                                                        {
                                                                        value = day + 31+29+31+30+31+30+31+31+30;
                                                                        }
                                                                        if(month==11)
                                                                                {
                                                                                value = day + 31+29+31+30+31+30+31+31+30+31;
                                                                                }
                                                                                if(month==12)
                                                                                        {
                                                                                        value = day + 31+29+31+30+31+30+31+31+30+31+30;
                                                                                        }
         }
```

```
else
        {
        if(month==3)
                {
                value = day + 31 + 28;
                }
        if(month==4)
                {
                value = day + 31+ 28 + 31;
                }
        if(month==5)
                {
                value = day + 31+ 28 + 31 +30;
                }
        if(month==6)
                {
                value = day + 31+ 28 + 31 +30 + 31;
                }

        if(month==7)
                {
                value = day + 31+ 28 + 31+30+31+30;
                }
        if(month==8)
                {
                value = day + 31+ 28 + 31+30+31+30+31;
                }
        if(month==9)
                {
                value = day + 31+28+31+30+31+30+31+31;
                }
        if(month==10)
                {
                value = day + 31+28+31+30+31+30+31+31+30;
                }
        if(month==11)
                {
                value = day + 31+28+31+30+31+30+31+31+30+31;
                }
        if(month==12)
                {
                value = day + 31+28+31+30+31+30+31+31+30+31+30;
                }
        }
return value;
}


//Function which takes an input and checks its of the correct length before allowing it to be passed to the validation function
int Buffer(char buffer[30], int len)
{
int length;
int valid=0;
```

```
length=strlen(buffer);

if(length==len)
        {
        valid=1;
        }
else
        {
        cout<<"\nPlease ensure that the input is of the appropriate length.";
        valid=0;
        }

getch();
return valid;
}
```

```
//---------------------------------------------------------------------------
/*****************************************************************************
Program: Installation
File: Installation.cpp
Functions: main
Description:Creates all files
Author: Fion McReynolds
Environment: Borland C++ Pro 6.0
Notes:
Revisions: 08/02/2024
*****************************************************************************/
#include<string.h>
#include<fstream.h>
#include<stdlib.h>
#include<math.h>
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<vcl.h>
#pragma hdrstop

//---------------------------------------------------------------------------

#pragma argsused

//CUSTOMER
char FileName2[80] = "CustomerFileSD";
typedef struct tag_cr{
        char title[10];
        char fnamecust[15];
        char lnamecust[15];
        char oneadcust[30];
        char twoadcust[30];
        char threeadcust[15];
        char pcodecust[9];
        char telnocust[12];
        char flag[2];
}CUSTOMER_RECORD;
int custref;
CUSTOMER_RECORD a_cust;

//INVOICE
char FileName5[80] = "InvoiceFileSD";
char invoiceref[10][3];
char custnum[10][3];
char quotenum[10][3];
char invoicedate[10][11];
char jobstartdate[10][11];
char jobenddate[10][11];
char paid[10][2];
int nii;
char nic[3];

//QUOTES
char FileName4[80] = "QuotesFileSD";
```

```
char quoteref[10][3];
char custno[10][3];
char quotedate[10][11];
char mainjobdesc[10][50];
char numofdays[10][3];
char labourq[10][5];
char mileage[10][4];
char vat[10][5];
char stockcost[10][5];
char totalcost[10][5];
int nqi;
char nqc[3];

//STAFF
char FileName1[80] = "StaffFileSD";
char staffref[10][3];
char fnamestaff[10][15];
char lnamestaff[10][15];
char oneadstaff[10][30];
char twoadstaff[10][30];
char threeadstaff[10][30];
char pcodestaff[10][9];
char telnostaff[10][12];
char emtel[10][12];
char ninum[10][15];
char username[10][15];
char password[10][15];
char loa[10][2];
int nsi;
char nsc[3];

//STOCK
char FileName3[80] = "StockFileSD";
char stockref[10][10];
char quantity[10][10];
char colour[10][15];
char volume[10][4];
char type[10][10];
char stockprice[10][8];
int nsti;
char nstc[3];

//LINKS
char FileName6[80] = "LinksFile";
char linksquoteref[10][3];
char linksstockref[10][3];
char linksquantity[10][4];
int nli;
char nlc[3];

//SCHEDULE
char FileName7[80]= "ScheduleFileSD";
char booking[3][370][14][3];
int staff;
int date;
```

```
int hour;

int Customer();
int Invoice();
int Staff();
int Quotes();
int Stock();
int Links();
int Schedule();
int ReWriteScheduleFile();

int main(int argc, char* argv[])
{
Customer();
Invoice();
Staff();
Quotes();
Stock();
Links();
Schedule();
return 0;
}
//CUSTOMER
int Customer()
{
getch();
ofstream fout(FileName2, ios::binary);
strcpy(a_cust.flag,"0");
for(custref=0;custref<10;custref++)
        {
        fout.write((char*)&a_cust,sizeof(a_cust));
        }//endfor
fout.close();
return 0;
}


//INVOICE
int Invoice()
{
int count;
nii=0;
itoa(nii,nic,10);
ofstream fout(FileName5, ios::binary);
fout.write((char*)&nic,sizeof(nic));
fout.close();
return 0;
}

//QUOTES
int Quotes()
{
int count;
nqi=0;
itoa(nqi,nqc,10);
```

```
ofstream fout(FileName4, ios::binary);
fout.write((char*)&nqc,sizeof(nqc));
fout.close();
return 0;
}

//STAFF
int Staff()
{
int count;
nsi=0;
itoa(nsi,nsc,10);
ofstream fout(FileName1, ios::binary);
fout.write((char*)&nsc,sizeof(nsc));
fout.close();
return 0;
}


//STOCK
int Stock()
{
int count;
nsti=0;
itoa(nsti,nstc,10);
ofstream fout(FileName3, ios::binary);
fout.write((char*)&nstc,sizeof(nstc));
fout.close();
return 0;
}

//LINKS
int Links()
{
int count;
nli=0;
itoa(nli,nlc,10);
ofstream fout(FileName6, ios::binary);
fout.write((char*)&nlc,sizeof(nlc));
fout.close();
return 0;
}

//SCHEDULE
int Schedule()
{
for(staff=0;staff<2;staff++)
        {
        for(date=0;date<366;date++)
                {
                for(hour=0;hour<13;hour++)
                        {
                        strcpy(booking[staff][date][hour],"*");
                        }//endfor - hour
                }//endfor - date
```

```
        }//endfor - staff
ReWriteScheduleFile();
return 0;
}

int ReWriteScheduleFile()
{
ofstream fout(FileName7, ios::binary);
for(staff=0;staff<2;staff++)
        {
        for(date=0;date<366;date++)
                {
                for(hour=0;hour<13;hour++)
                        {
                        fout.write((char*)&booking[staff][date][hour],1);
                        fout.write("\n",1);
                        }//endfor - hour
                }//endfor - date
        }//  endfor - staff
getch();
fout.close();
return 0;
}
//---------------------------------------------------------------------------
```