Problems and sub problems

Objective	Problems	Sub Problems	Justification
Allow the user to add a new employee to the system, entering key information including: Full name, Address, Telephone number, national insurance number, username, password and level of access. This will then be stored in a permanent file.	Produce the read back and read write routines for the files. Include the 'Add Staff' to the menu structure. Use a program to input the required information. Validation on all inputs.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read.	I have broken down the problem in this way to allow the routines created to be used throughout the program for adding to a file. I have also chosen to break the problem down to make the menu structure easy to read so it can be reused throughout the system and consistent for the staff when using it.
Permit the user to change information stored on an employee e.g., Address and telephone number.	Produce the read back and read write routines for the files. Include the 'Change Information' to the menu structure. Use a program to input the required information. Re-validate the information entered.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read. Search to find the information you wish to change exists.	I have broken down the problem in this way to allow the routines created to be used throughout the program for changing in a file. I have also chosen to break the problem down to make the menu structure easy to read so it can be reused throughout the system and consistent for the staff when using it. Furthermore, I chose to break down the problem to search for the information to reduce time-wasting if the staff member no longer exists.

Ability to delete a member of staff that no longer works there.	Produce the read back and read write routines for the files. Include the 'Delete Staff Member' to the menu structure. Make sure the staff member you wish to delete exists on the system before attempting to remove them.	Make sure that the menu structure and delete section is clear and easy to understand. Use validation routine to ensure the staff member exists by checking within the staff file.	I have broken down the problem in this way to allow the routines created to be used throughout the program for deleting from a file. I have also chosen to break the problem down to make the menu structure easy to read so it can be reused throughout the system and consistent for the staff when using it.
Allow the user to search for staff via staff reference.	Produce a routine which will search for staff members by staff reference. Create a routine to read back the file.	Make sure the member of staff exists. Produce a nicely formatted list of staff information.	I have broken down the problem in this way to ensure the validation that the item searched for exists and the readback routine can be re-used in other searches.
Include validation on all data entered when adding a new employee.	Validate the following information: - Postcode - National Insurance number - Telephone Number	Produce routines to validate the information using parameter passing.	I have broken down the problem in this way to allow the validation routines to be independent so they can be re used in other areas of the program.
Ensure all information saved is backed up.	Create a routine to allow the backing up of data. Produce read back and rewrite routines for the file. Use security to allow only certain staff to back up data.	Produce validation routine to ensure the level of access before allowing data to be backed up.	I have broken down the problem in this way to allow the validation routine for security to be reused.
Allow the user to enter a new customer to the system by storing: Full name, Address, and telephone number in a permanent file.	Produce the read back and read write routines for the files. Include the 'Add Customer' to the menu structure. Use a program to input the required information. Validation on all inputs.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read.	I have broken down the problem in this way to allow the routines created to be used throughout the program for adding to a file. I have also chosen to break the problem down to make the

			menu structure easy to read so it can be reused throughout the system and consistent for the staff when using it.
Allow the user to delete a customer from the system when they are no longer using the company.	Produce the read back and read write routines for the files. Include the 'Delete Customer' to the menu structure. Make sure the customer you wish to delete exists on the system before attempting to remove them.	Make sure that the menu structure and delete section is clear and easy to understand. Use validation routine to ensure the customer exists by checking within the customer file.	I have broken down the problem in this way to allow the routines created to be used throughout the program for deleting from a file. I have also chosen to break the problem down to make the menu structure easy to read so it can be reused throughout the system and consistent for the staff when using it.
Have the ability to change the address or telephone number of a customer stored on the system.	Produce the read back and read write routines for the files. Include the 'Change Information' to the menu structure. Use a program to input the required information. Re-validate the information entered.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read. Search to find the information you wish to change exists.	I have broken down the problem in this way to allow the routines created to be used throughout the program for changing in a file. I have also chosen to break the problem down to make the menu structure easy to read so it can be reused throughout the system and consistent for the staff when using it. Furthermore, I chose to break down the problem to search for the information to reduce time-wasting if the customer no longer exists.
Enable the user to search for customer using name or cstomer reference.	Produce a routine which will search for customers by name or by reference. Create a routine to read back the file.	Make sure the customer exists. Produce a nicely formatted list of customer information.	I have broken down the problem in this way to ensure the validation that the item searched for exists and the readback routine can be re-used in other searches.

Ensure all data entered is validated.	Validate the following information: - Postcode - Telephone number	Produce routines to validate the information using parameter passing.	I have broken down the problem in this way to allow the validation routines to be independent so they can be re used in other areas of the program.
Ensure all information saved is backed up.	Create a routine to allow the backing up of data. Produce read back and rewrite routines for the file. Use security to allow only certain staff to back up data.	Produce validation routine to ensure the level of access before allowing data to be backed up.	I have broken down the problem in this way to allow the validation routine for security to be reused.
Allow the user to add a new quote to the system by storing it in a permanent file. The information to be stored on a quote will be: Quote reference, customer reference, job description, expected number of days on the job and travel costs	Produce the read back and read write routines for the files. Include the 'Add Quote' to the menu structure. Use a program to input the required information. Validation on all inputs.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read.	I have broken down the problem in this way to allow the routines created to be used throughout the program for adding to a file. I have also chosen to break the problem down to make the menu structure easy to read so it can be reused throughout the system and consistent for the staff when using it.
Enable the user to delete a quote once the booking is finalised.	Produce the read back and read write routines for the files. Include the 'Delete Quote' to the menu structure. Make sure the quote you wish to delete exists on the system before attempting to remove it.	Make sure that the menu structure and delete section is clear and easy to understand. Use validation routine to ensure the quote exists by checking within the quotes file.	I have broken down the problem in this way to allow the routines created to be used throughout the program for deleting from a file. I have also chosen to break the problem down to make the menu structure easy to read so it can be reused throughout the system and consistent for the staff when using it.

The ability to change the information on the quote including the estimated price and the number of days required.	Produce the read back and read write routines for the files. Include the 'Change Information' to the menu structure. Use a program to input the required information. Re-validate the information entered.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read. Search to find the information you wish to change exists.	I have broken down the problem in this way to allow the routines created to be used throughout the program for changing in a file. I have also chosen to break the problem down to make the menu structure easy to read so it can be reused throughout the system and consistent for the staff when using it. Furthermore, I chose to break down the problem to search for the information to reduce time-wasting if the quote no longer exists.
Enable the user to search for a specific quote using quote reference or customer name.	Produce a routine which will search for a quote by quote reference or customer name. Create a routine to readback the file.	Make sure the quote exists. Produce a nicely formatted list of details of the quote.	I have broken down the problem in this way to ensure the validation that the item searched for exists and the readback routine can be re-used in other searches.
Produce a calculation for the quote taking into account current stock prices, labour costs, number of days worked and mileage.	Produce a multistage calculation using different mathematical operators and parameter passing to return the value of the calculation.	Produce a format to show the stages of the calculation broken down including VAT.	I have broken down the problem in this way to allow the stages of the calculation to be clear and easy to break down for the member of staff when entering the information.
Validate all quote information upon entry.	Validate the following information: - Quote reference	Produce routines to validate the information using parameter passing.	I have broken down the problem in this way to allow the validation routines to be independent so they can be re used in other areas of the program.
Ensure all information saved is backed up.	Create a routine to allow the backing up of data. Produce read back and rewrite routines for the file.	Produce validation routine to ensure the level of access before allowing data to be backed up.	I have broken down the problem in this way to allow the validation routine for security to be reused.

	Use security to allow only certain staff to back up data.		
Allow the user to add additional stock to a permanent file including information such as: stock ID, colour of paint, type of paint, volume of can, price and quantity.	Produce the read back and read write routines for the files. Include the 'Add Stock' to the menu structure. Use a program to input the required information. Validation on all inputs.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read.	I have broken down the problem in this way to allow the routines created to be used throughout the program for adding to a file. I have also chosen to break the problem down to make the menu structure easy to read so it can be reused throughout the system and consistent for the staff when using it.
Allow the user to delete stock from the system if it is no longer required or is no longer sold.	Produce the read back and read write routines for the files. Include the 'Delete Item of Stock' to the menu structure. Make sure the item of stock you wish to delete exists on the system before attempting to remove it.	Make sure that the menu structure and delete section is clear and easy to understand. Use validation routine to ensure the item of stock exists by checking within the stock file.	I have broken down the problem in this way to allow the routines created to be used throughout the program for deleting from a file. I have also chosen to break the problem down to make the menu structure easy to read so it can be reused throughout the system and consistent for the staff when using it.
The ability to change the price and quantity of the stock.	Produce the read back and read write routines for the files. Include the 'Change Information' to the menu structure. Use a program to input the required information. Re-validate the information entered.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read. Search to find the information you wish to change exists.	I have broken down the problem in this way to allow the routines created to be used throughout the program for changing in a file. I have also chosen to break the problem down to make the menu structure easy to read so it can be reused throughout the system and consistent for the staff when using it. Furthermore, I chose to break down the

			problem to search for the information to reduce time-wasting if the quote no longer exists.
Enable the user to search for stock using stock ID to display information.	Produce a routine which will search for stock by stock ID. Produce a routine to read back the file.	Make sure the item of stock exists. Produce a nicely formatted list of stock information.	I have broken down the problem in this way to ensure the validation that the item searched for exists and the readback routine can be re-used in other searches.
Enable the user to sort stock quantities from low to high.	Produce a routine which will sort stock quantities. Have the option to sort stock on the user interface.	Make sure there is stock to sort. Create a program to read back and rewrite to the file.	I have broken down the problem like this so that the sort can be a recursive algorithm.
Produce a message when stock quantities drop to a specific level.	Carry out a loop which will produce a message when stock reaches a certain level.	Make sure the loop will stop running after it has checked through all the stock and only produce the message when there is stock below the quantity specified.	I have broken down the problem like this so that the loop isn't continuous and will stop after a certain point.
Ensure information entered is validated.	Validate the following information: - Price - Quantity	Produce routines to validate the information using parameter passing.	I have broken down the problem in this way to allow the validation routines to be independent so they can be re used in other areas of the program.
Ensure all information saved is backed up.	Create a routine to allow the backing up of data. Produce read back and rewrite routines for the file. Use security to allow only certain staff to back up data.	Produce validation routine to ensure the level of access before allowing data to be backed up.	I have broken down the problem in this way to allow the validation routine for security to be reused.
Allow the user to add a new booking to the schedule including: staff reference, quote reference and dates of the job and then stores this in a permanent text file.	Produce the read back and read write routines for the files. Include the 'Add to schedule' to the menu structure.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read.	I have broken down the problem in this way to allow the routines created to be used throughout the program for adding to a file.

	Use a program to input the required information. Validation on all inputs.		I have also chosen to break the problem down to make the menu structure easy to read so it can be reused throughout the system and consistent for the staff when using it.
Allow only those with the highest access to change information such as date of the job.	Produce the read back and read write routines for the files. Include the 'Change Information' to the menu structure. Use a program to input the required information. Re-validate the information entered. Use validation to check the access level before allowing the change to happen.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read. Search to find the information you wish to change exists. Produce a message if the level of access is invalid.	I have broken down the problem in this way to allow the routines created to be used throughout the program for changing in a file. I have also chosen to break the problem down to make the menu structure easy to read so it can be reused throughout the system and consistent for the staff when using it. Furthermore, I chose to break down the problem to search for the information to reduce time-wasting if the booking no longer exists.
Have the ability to delete information such as a cancelled booking.	Produce the read back and read write routines for the files. Include the 'Remove Cancellation' to the menu structure. Make sure the booking you wish to delete exists on the system before attempting to remove it.	Make sure that the menu structure and delete section is clear and easy to understand. Use validation routine to ensure the quote exists by checking within the quote file.	I have broken down the problem in this way to allow the routines created to be used throughout the program for deleting from a file. I have also chosen to break the problem down to make the menu structure easy to read so it can be reused throughout the system and consistent for the staff when using it.
Include validation on the new information which is entered including the date.	Validate the following information: - Date - Quote reference - Staff number	Produce routines to validate the information using parameter passing.	I have broken down the problem in this way to allow the validation routines to be independent so they

			can be re used in other areas of the program.
Ensure there is hierarchical access with levels of access of the administrator and the members of staff below.	Produce a program using parameter passing to check whether a staff member has the ability to access and edit the schedule.	Create messaged to be outputted to the screen to tell a user if they do not have access.	
Ensure all information saved is backed up.	Create a routine to allow the backing up of data. Produce read back and rewrite routines for the file. Use security to allow only certain staff to back up data.	Produce validation routine to ensure the level of access before allowing data to be backed up.	I have broken down the problem in this way to allow the validation routine for security to be reused.
Allow the user to add a new invoice to the system storing this in a permanent text file including the following information: Invoice reference, customer reference, dates of the job, job description, number of days worked, cost of materials, mileage costs and payment status..	Produce the read back and read write routines for the files. Include the 'Add Invoice' to the menu structure. Use a program to input the required information. Validation on all inputs.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read.	I have broken down the problem in this way to allow the routines created to be used throughout the program for adding to a file. I have also chosen to break the problem down to make the menu structure easy to read so it can be reused throughout the system and consistent for the staff when using it.
Allow the user to be able to change whether the job has been paid for.	Produce the read back and read write routines for the files. Include the 'Change Information' to the menu structure. Use a program to input the required information. Re-validate the information entered.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read. Search to find the information you wish to change exists.	I have broken down the problem in this way to allow the routines created to be used throughout the program for changing in a file. I have also chosen to break the problem down to make the menu structure easy to read so it can be reused throughout the system and consistent for the staff when using it. Furthermore, I chose to break down the

			problem to search for the information to reduce time-wasting if the invoice no longer exists.
The ability to delete an invoice from the system once it is no longer needed and the job has been paid.	Produce the read back and read write routines for the files. Include the 'Delete Invoice' to the menu structure. Make sure the invoice you wish to delete exists on the system / that the file is not empty before attempting to remove it.	Make sure that the menu structure and delete section is clear and easy to understand. Use validation routine to ensure the invoice exists by checking within the invoices file.	I have broken down the problem in this way to allow the routines created to be used throughout the program for deleting from a file. I have also chosen to break the problem down to make the menu structure easy to read so it can be reused throughout the system and consistent for the staff when using it.
Produce a multistage calculation for the invoice taking into account: number of days worked, mileage costs, current stock costs and labour costs.	Produce a multistage calculation using different mathematical operators and parameter passing to return the value of the calculation.	Produce a format to show the stages of the calculation broken down including VAT.	I have broken down the problem in this way to allow the stages of the calculation to be clear and easy to break down for the member of staff when entering the information.
Enable the user to be able to search for an invoice by name.	Produce a routine which will search for invoices by name. Use a readback routine.	Make sure the invoice exists. Produce a nicely formatted list displaying details of an invoice.	I have broken down the problem in this way to ensure the validation that the item searched for exists and the readback routine can be re-used in other searches.
Enable the user to be able to search for paid or unpaid invoices.	Produce a routine which will search for the invoices by whether they have been paid or not. Use a readback routine.	Produce a list of invoices that are currently unpaid.	I have broken down the problem in this way to help clearly define which are unpaid by verifying the information stored in file using the readback routine which can then be used over all the searches.
Ensure there is validation in place for all information entered excluding the description of the job.	Validate the following information: - Invoice reference - Staff number	Produce routines to validate the information using parameter passing.	I have broken down the problem in this way to allow the validation routines to be independent so they

	- Price - Customer reference		can be re used in other areas of the program.
Ensure all information saved is backed up.	Create a routine to allow the backing up of data. Produce read back and rewrite routines for the file. Use security to allow only certain staff to back up data.	Produce validation routine to ensure the level of access before allowing data to be backed up.	I have broken down the problem in this way to allow the validation routine for security to be reused.

Objective	Outputs
Allow the user to add a new employee to the system, entering key information including: Full name, Address, Telephone number and national insurance number. This will then be stored in a permanent file.	N/A
Permit the user to change information stored on an employee e.g., Address and telephone number.	N/A
Ability to delete a member of staff that no longer works there.	N/A
Allow the user to search for staff via reference.	Display of staff information including: Name, Home Address, Telephone number and National Insurance Number.
Include validation on all data entered when adding a new employee.	See error messages from validation design.
Ensure all information saved is backed up.	N/A
Allow the user to enter a new customer to the system by storing: Full name, Address, Telephone number, and a basic description of the job in a permanent file.	N/A
Allow the user to delete a customer from the system when they are no longer using the company.	N/A
Have the ability to change the address or telephone	N/A

number of a customer stored on the system.	
Enable the user to search for customer using name or customer reference.	Display of the following customer information: Name, Home Address, Telephone number and a basic job description.
Ensure all data entered is validated.	See error messages from validation design.
Ensure all information saved is backed up.	N/A
Allow the user to add a new quote to the system by storing it in a permanent file. The information to be stored on a quote will be: quote reference, customer reference, job description, expected number of days on the job and travel costs.	N/A
Enable the user to delete a quote once the booking is finalised.	N/A
The ability to change the information on the quote including the estimated price and the number of days required.	N/A
Enable the user to search for a specific quote using quote reference or customer name.	Display the following information on a quote: Customer reference, customer name, customer telephone number, job description, number of days worked and price breakdown.
Produce a calculation for the quote taking into account current stock prices, labour costs, number of days worked and mileage.	Display the cost of each of the following sections: Material costs, Labour costs, mileage and VAT. Also display the total cost at the end.

Validate all quote information upon entry.	See error messages from validation design.
Ensure all information saved is backed up.	N/A
Allow the user to add additional stock to a permanent file including information such as: colour of paint, type of paint, volume of can, price and quantity.	N/A
Allow the user to delete stock from the system if it is no longer required or is no longer sold.	N/A
The ability to change the price and quantity of the stock.	N/A
Enable the user to search for stock using stock ID to display information.	Display the following stock information: Colour, price and quantity.
Enable the user to sort stock quantities from low to high.	Display the stock reference and the quantity of the stock in the sorted list.
Produce a message when stock quantities drop to a specific level.	Display a simple output message when stock drops so low.
Ensure information entered is validated.	See error messages from validation design.
Ensure all information saved is backed up.	N/A
Allow the user to add a new booking to the schedule including: Quote reference and the dates of the job and stores this in a permanent text file.	N/A
Allow only those with the highest access to change information such as date of the job.	N/A
Have the ability to delete information such as a cancelled booking.	N/A
Allow a user to view the schedule of one staff member at a time	Display the following information: Dates, times and quote reference.

Include validation on the new information which is entered including the date.	See error messages from validation design.
Ensure there is hierarchical access with levels of access of the administrator and the members of staff below.	See in validation.
Ensure all information saved is backed up.	N/A
Allow the user to add a new invoice to the system storing this in a permanent text file including the following information: invoice reference, customer reference, dates of the job, job description, number of days worked, cost of materials, travel costs and payment status.	N/A
Allow the user to be able to change information including whether the job has been paid for.	N/A
The ability to delete an invoice from the system once it is no longer needed and the job has been paid.	N/A
Produce a multistage calculation for the invoice taking into account: number of days worked, mileage costs, current stock costs and labour costs.	Display the cost of each of the following sections: Material costs, Labour costs, mileage and VAT. Also display the total cost at the end.
Enable the user to be able to search for an invoice by invoice reference.	For each invoice searched display: Customer name, customer reference, customer telephone number, job descriptions price breakdown, and dates of the job.
Enable the user to be able to search for paid or unpaid invoices.	Display the customer reference and the status of the payment.
Ensure there is validation in place for all information	See error messages from validation design.

entered excluding the description of the job.	
Ensure all information saved is backed up.	N/A

Objective	Inputs
Allow the user to add a new employee to the system, entering key information. This will then be stored in a permanent file.	Required inputs will be displayed for the following information: Full name, Address, Telephone number, national insurance number, username, password and level of access.
Allow the user to enter a new customer to the system, entering key information. This will then be stored in a permanent file.	Required inputs will be displayed for the following information: Full name, Address and Telephone number.
Allow the user to add a new quote to the system, entering key information. This will then be stored in a permanent file.	Required inputs will be displayed for the following information: Quote reference, customer reference, job description, expected number of days on the job and travel costs.
Allow the user to add stock to the system, entering key information. This will then be stored in a permanent file.	Required inputs will be displayed for the following information: Colour of the paint, type of paint, volume of the can, price and quantity.
Allow the user to add a new booking to the schedule, entering key information. This will then be stored in a permanent file.	Required inputs will be displayed for the following information: Quote reference and the dates of the job.
Allow the user to add an invoice to the system, entering key information. This will then be stored in a permanent file.	Required inputs will be displayed for the following information: Invoice reference, customer reference, dates of the job, job description, number of days worked, cost of materials, mileage costs and payment status.

Staff

1. Add staff
2. Change staff information
3. Delete staff
4. View staff
5. Return to main menu

Change staff information

1. Change home address
2. Change telephone number
3. Return to staff menu

Main Menu:

1. Staff
2. Customers
3. Quotes
4. Invoices
5. Stock
6. Schedule
7. Quit

Customers

1. Add customer
2. Change customer information
3. Delete customer
4. View customer
5. Return to main menu

Change customer information

1. Change home address
2. Change telephone number
3. Return to customer menu

View Customer

1. Search by name
2. Search by customer reference
3. Return to customer menu

Quotes

1. Add quote
2. Change quote information
3. Delete quote
4. View quote
5. Return to main menu

Change quote information

1. Change quote price
2. Change number of days required
3. Return to quote menu

View Quote

1. Search by quote reference
2. Search by customer name
3. Return to quote menu

Invoices

1. Add invoice
2. Change invoice information
3. Delete invoice
4. View invoice
5. Return to main menu

Change invoice information

1. Change dates worked
2. Change payment status
3. Return to invoice menu

View Invoice

1. Search by customer name
2. Search for unpaid invoices
3. Search for paid invoices
4. Return to invoice menu

Stock

1. Add stock
2. Change stock information
3. Delete stock
4. View stock
5. Return to main menu

Change stock information

1. Change price of stock
2. Change stock quantities
3. Return to stock menu

View Stock

1. Search by stock ID
2. Sort stock quantity low to high
3. Return to stock menu

Schedule

1. Add booking
2. Change booking information
3. Delete booking
4. View schedule
5. Return to main menu

View Schedule

1. View staff 1
2. View staff 2
3. Return to schedule menu

Grids

The following document shows how the inputs and outputs would occur logically within the system.

Text size = 12 , Colour = Black background, white text , font = consolas → same throughout

Main Menu
****

1. Staff
2. Customers
3. Quotes
4. Invoices
5. Stock
6. Schedule
7. Admin
8. Quit

Enter choice:

Staff member inputs information about a new employee; must have correct hierarchical access

```
Add Staff
*********

Enter new staff reference: 1

Enter first name: Steve
Enter last name : Russel

Enter address line 1: 42 Fir Tree Drive
Enter address line 2: Wales
Enter aaaress line 3: Sheffield
Enter postcode      : S26 5kZ

Enter mobile number: 07796193489
Enter national insurance number: AB 123456 Z

Enter username: S.Russel
Enter password: Russe!42
Enter level of access: 3
```

Each staff reference will require a unique identifier on the system

In case any staff or customers need to contact the employee

Required to secure details on an employee at a company

Unique login to the system with hierarchical access so staff can only access what they need to, to increase security.

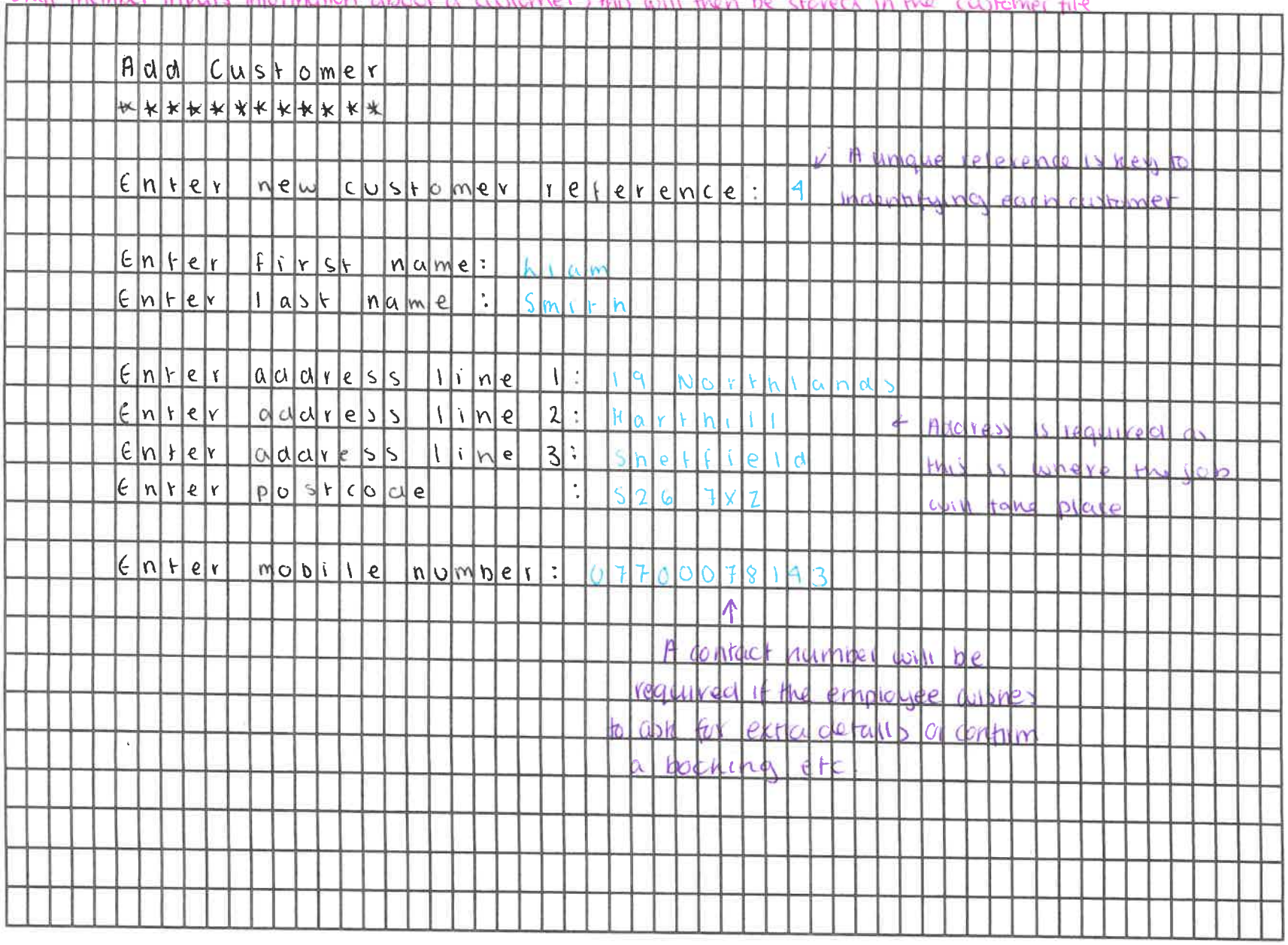
A search which outputs important information on a specific staff member

```
View Staff Member by Staff Reference
**********************************

Enter staff reference: 2

Name: Stuart Watson

Aadress: 12 Oak Drive
         Kiveton Park
         Sheffield
         S26 5SQ

Telephone Number: 07719890314

NI Number: QQ 123456 C

Level of access: 3
```

Important to output in case the member of staff needs to be contacted immediately

A requirement have stored the national insurance number of an employee

Staff member inputs information about a customer, this will then be stored in the customer file



A search which outputs customer information, using the customer name to find them in the file

View Customer by Customer Name
****

Enter customers last name: Smith

Customer reference: 4

Address line 1: 19 Northlands
Address line 2: Hartnill
Address line 3: Sheffield
Postcode : S26 7XZ

← Staff will need the address as that is where the job will be

Telephone number: 07700078143

↑ In case the staff need to contact the customer regarding the job

A search which outputs information on a customer by using the reference to find information in the file

View Customer by Customer Reference
****

Enter customer reference: 4

Customer name: Liam Smith

Address line 1: 19 Northlands ← So the staff know where the job will be when they are going
Address line 2: Hartnill
Address line 3: Sheffield
Postcode : S26 7XZ

Telephone number: 07700078143
↑ In case the staff need to contact the customer about the job

Staff member enters information on a quote; Stock price is linked automatically from the file to be used in the calculation.

Add Quote
****

Unique ID to identify each quote

Enter quote reference: 4
Enter customer reference: 4 ← Helps link to customer

Job description: PUT
KITCHEN
CRM

← Helps the staff remember the details of the job as it gets closer to the time

Enter number of days on the job: 5
Enter travel costs: 60

How many items of stock are required: 4

Enter stock ID: 2
Enter stock ID: 1
Enter stock ID: 3
Enter stock ID: 5

← So the system can find the price of each item of stock to be used in the calculation

Price Breakdown
****

Materials	:	150
Labour	:	450
Mileage	:	60
VAT	:	132
Total	:	792

← so the customer has an expectation on what the price will be

A search which uses a unique quote reference entered by a member of staff to view a quote

```
View Quote by Quote Reference
*****************************

Enter quote reference: 4

Customer name: Liam Smith
Telephone number: 07749173406
Customer reference: 4

Job description: PVT
                 KITCHEN
                 CRM

Number of days worked: 7

Price Breakdown
***************

Materials: 90
Labour   : 560
Mileage  : 25
VAT      : 135
Total    : 810
```

Outputted since the customer reference will help link to other aspects of the company, such as the invoice. Contact information allows staff to easily get in contact with the customer if something appears wrong/different on the quote

A price is calculated and outputted to the screen so the staff can let the customer know a price for the job

A search which uses the customer reference to read back against the file

```
View Quote by Customer Reference
********************************

Enter customer reference: 4

Customer name: Liam Smith
Telephone number: 07749173406
Quote reference: 4

Job description: PNT
                 KITCHEW
                 CRM

Number of days worked: 7

Price Breakdown
***************

Materials: 90
Labour   : 560
Mileage  : 25
VAT      : 135
Total    : 810
```

← To remind the staff of what they will be doing when they arrive at the job

← So the staff can plan what they will do each day on the job to make sure it is completed on time

← So the staff are able to answer any enquiries about the price of the job from the customer.

Staff member enters invoice information which will then be stored for each customer who books

```
Add invoice
**********

Enter invoice reference: 1
Enter customer reference: 4

Enter start date: 15/08/2023
Enter end date  : 22/08/2023

Enter job description: PNT
                       KITCHEN
                       CRM

Enter number of days worked: 7
Enter cost of materials    : 90
Enter travel costs         : 25

Is the job paid for: 1
(0 = yes, 1 = No)

Price Breakdown
***************
```

Unique to identify all invoices in the system

links to customer information

User inputs so the calculation can take place for the final price to tell the customer

Allows unpaid invoices to be found on the system so that the company can track customers that haven't paid yet

Materials	: 90
Labour	: 560
Mileage	: 25
VAT	: 135
Total	: 810

← Price breakdown so the staff can answer any customer enquiries about the cost

A search which uses the invoice reference inputted by the user, to view an invoice for a job.

```
View Invoice by Invoice Reference
*********************************

Enter invoice reference: 4

Customer reference: 4
Customer name: Liam Smith
Telephone number: 07491236455

Job description: PAT
                 KITCHEN
                 CRM

Dates worked: 05/08/2023-12/08/2023

Price Breakdown
***************

Materials:  90
Labour   :  560
Mileage  :  25
VAT      :  135
Total    :  810
```

Outputted since if a staff member wants to contact the customer regarding an invoice they can contact them quickly.

Dates are shown so both staff and customers can see the invoice and agree these are the days worked and these are the days which will be paid for

Price is outputted in a broken down format to help the customer know how they have been charged

A search which outputs which jobs currently haven't been paid for.

View unpaid invoices
****

Customer reference: 2

Customer reference: 4

Customer reference: 9

← Outputs the customer reference so that the staff can go and view a customer's information, such as a telephone number

Staff member inputs stock information which is then stored in the stock file

Add Stock
****

Enter new stock reference: 1 ← helps to identify each piece of stock individually.

Colour: White ← (when an employee searches for stock the customer will have requested colours and type) so the staff need to know exactly what they have in stock

Type of paint: Matt ←
(Use N/A if not paint)

Price: 15 ← used for the calculation

Volume: 10L

Quantity: 4 ← helps staff to know if they require more stock, or if there's enough for the job

A search which outputs specific information about a piece of stock, to help staff prepare for a job

View Stock by Stock Reference
****

Enter stock reference: 1

Colour: White ← Included colour as the customer will have requested it so the staff will only want to check for the colour they need

Price: 15

Quantity: 4

So the staff are prepared when they go to buy, but also to help prevent a quick / invoice

Checks quantity since if its 0 or 1 staff will need to go and buy some more to be prepared for jobs

A sort which shows the stock quantities in order by retrieving information from the stock file

```
Sort Stock Quantities
*********************

Stock reference: 1
Quantity       : 3

Stock reference: 5
Quantity       : 6

Stock reference: 7
Quantity       : 9
```

← By using stock reference it makes it very clear to the staff what is the lowest quantity and what is needed

← Quantity is shown because if there was lots of recent purchases, even though something may have the lowest quantity it could still be a high number

Staff member enters the booking information below to be added to the schedule

```
Add Booking
**********

Enter staff reference: 1
Enter number of days on the job: 2
Enter date: 17/08/2023
Enter number of hours you want to work that day: 4
Enter hour: 11
Enter hour: 12
Enter hour: 13
Enter hour: 14

Enter quote reference: 5

Enter date: 18/08/2023
Enter number of hours you want to work that day: 2
Enter hour: 7
Enter hour: 8

Enter quote reference: 5
```

← So the booking is added to the correct staff members schedule

← Used for inputting into the schedule so the staff can see when they're booked

← Used for inputting to the schedule to see what times of day staff are working

← Inputted so this is what's shown on the schedules to see when they are working

Outputs the 3D array to show a specific employees schedule

```
View Staff 1
**********

Staff 1:

    7  8  9  10 11 12 13 14 15 16 17 18 19
1   4  4  4  4  4
2
3
4           7  7  7  7  7  7  7  7
.
.
.
364
365
```

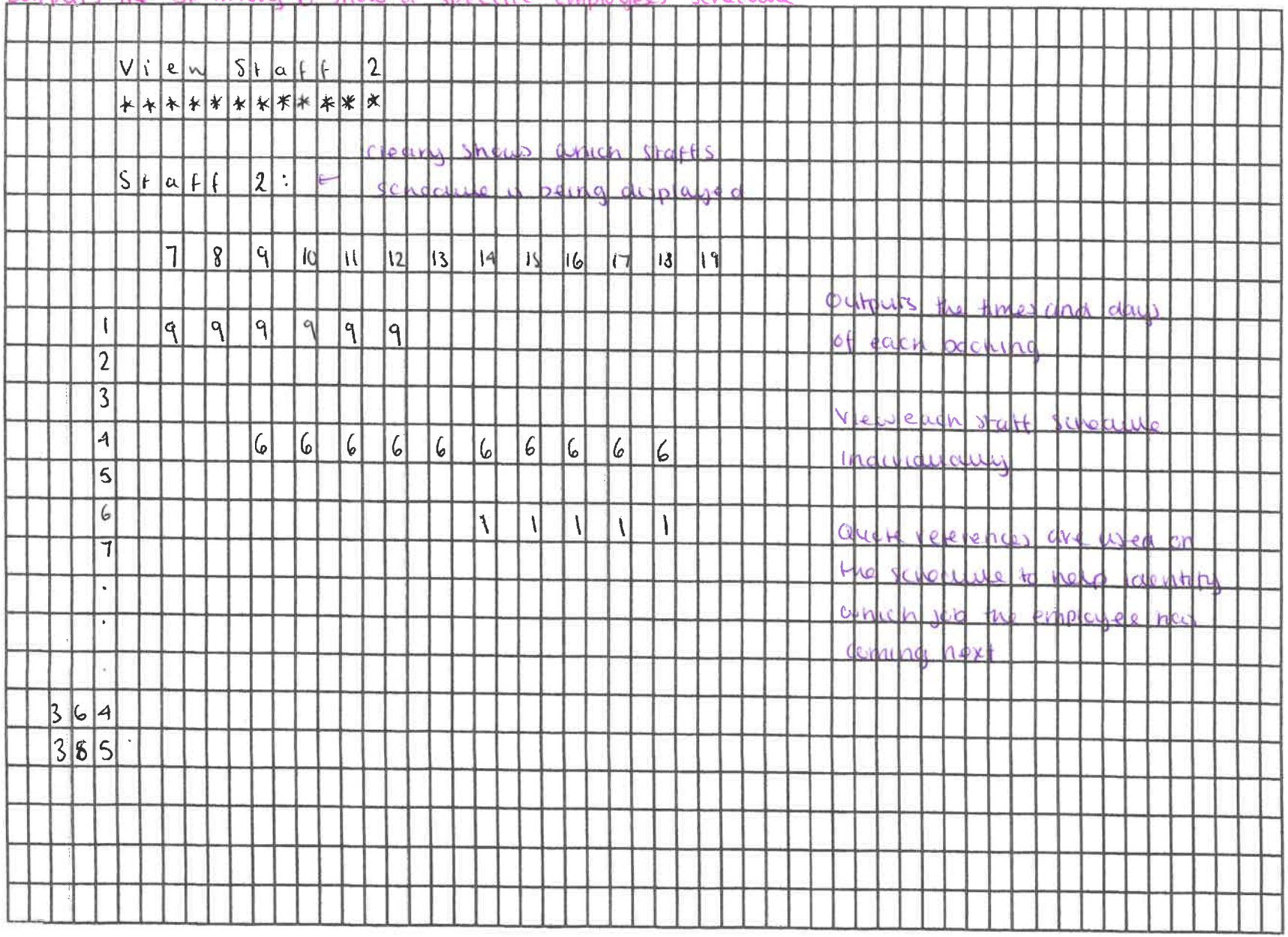
Clearly shows which schedule is outputted

Shows the time and days of each working

Viewing one schedule at a time

Quote references are used on the schedule to help identify which join the employee has coming next.

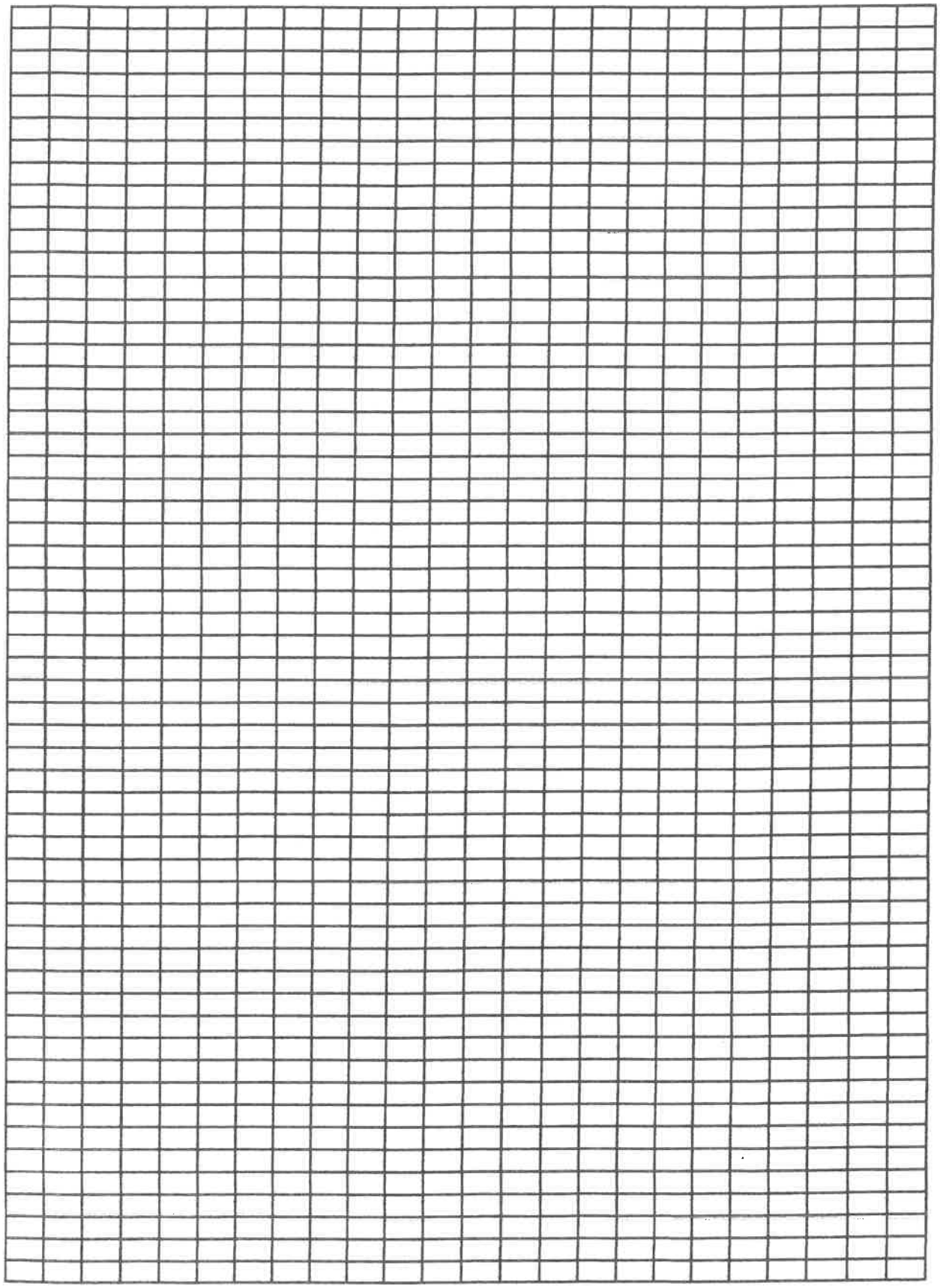
Outputs the 3D Array to show a specific employees schedule

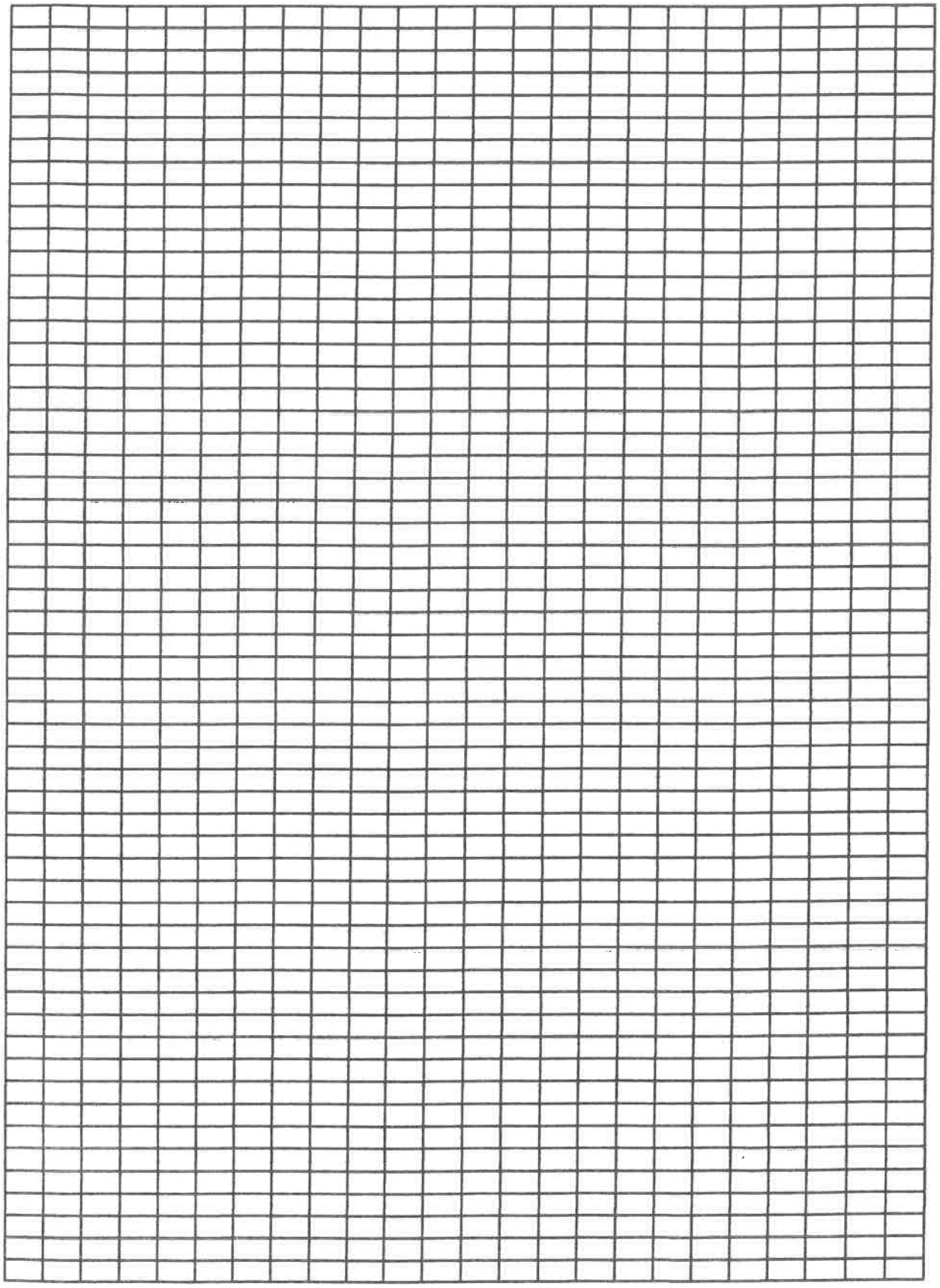


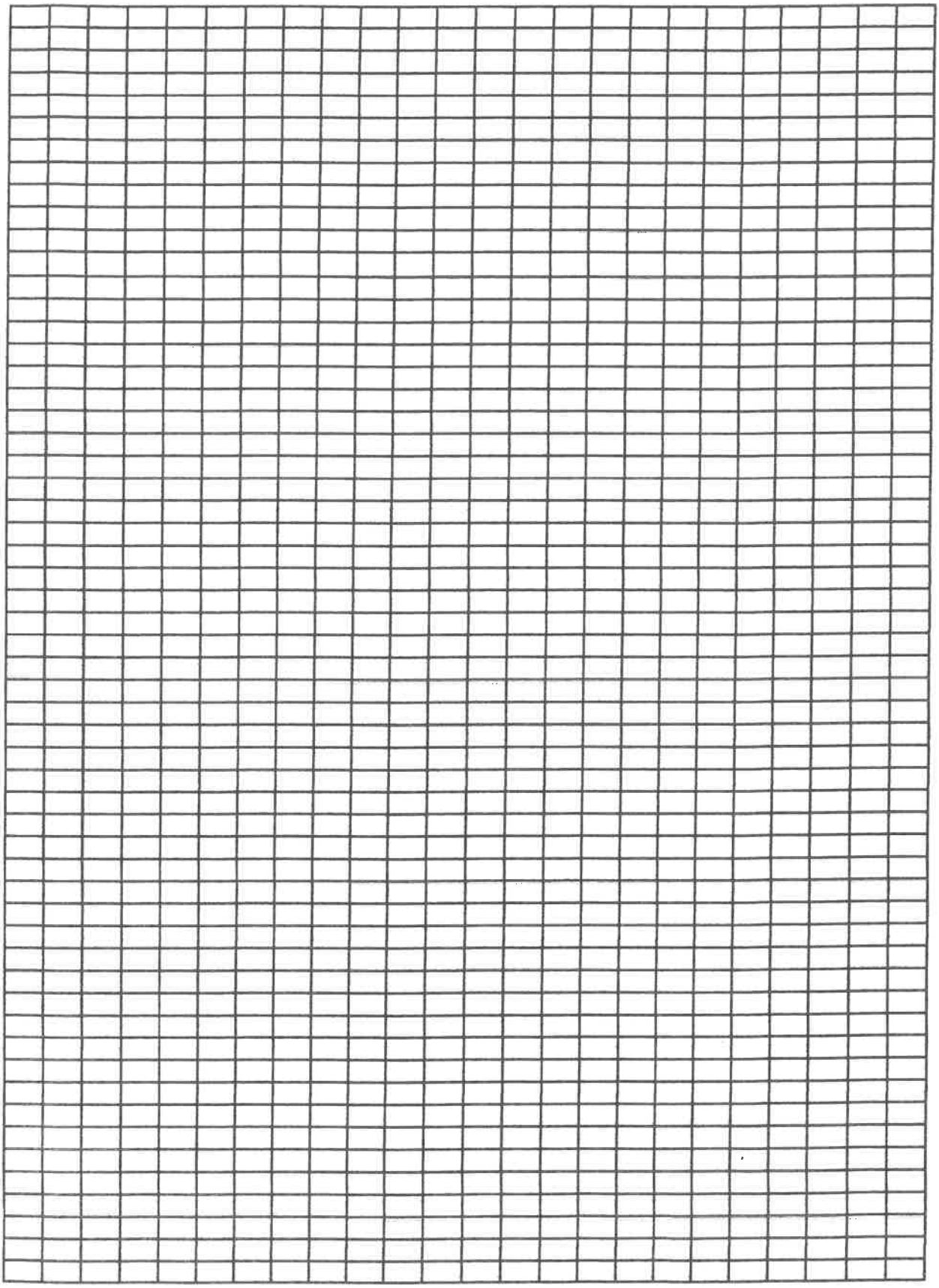
Outputs the times and days of each booking

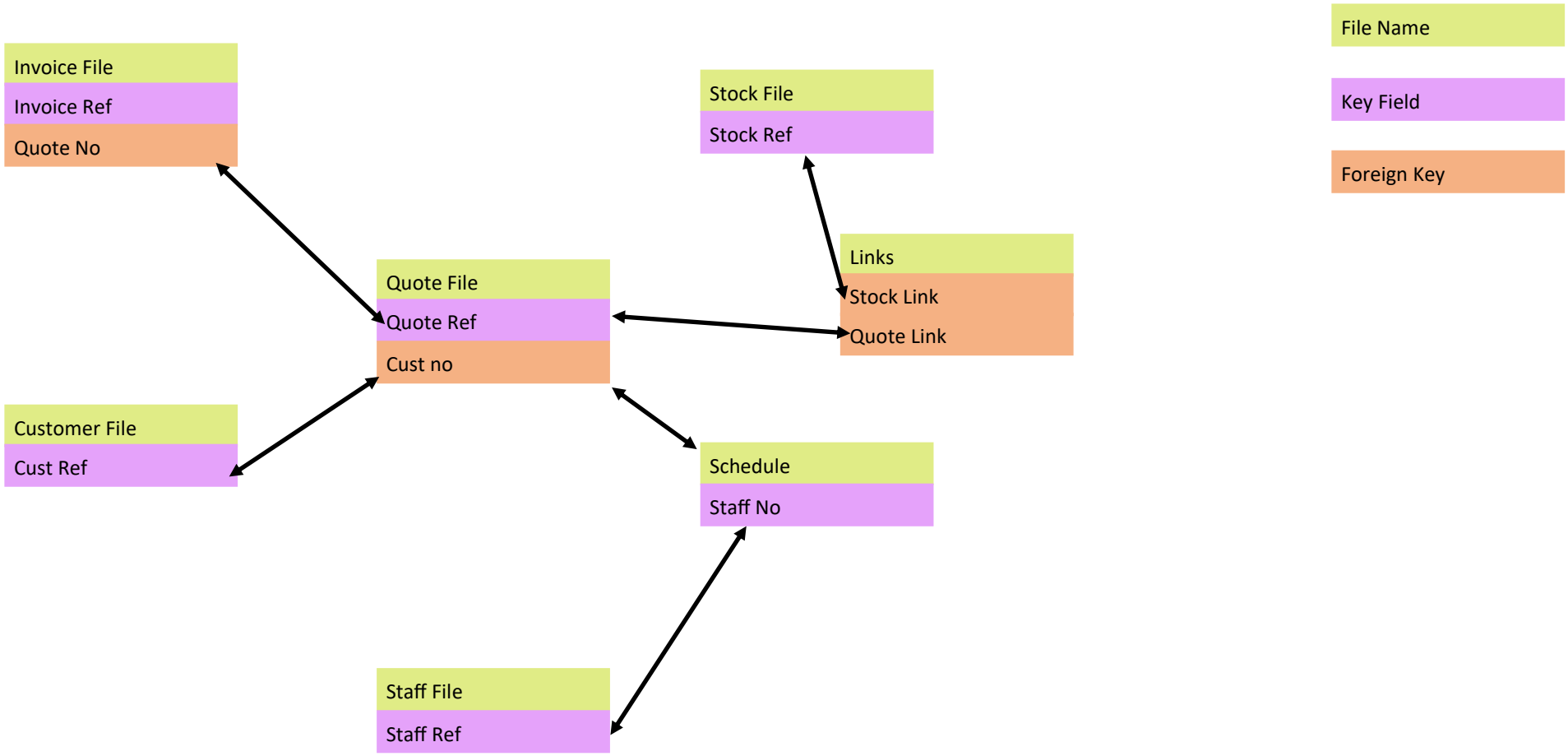
View each staff schedule individually

Quick references are given on the schedule to help identify which job the employee has coming next








File Name
Key Field
Foreign Key
Invoice File
Invoice Ref
Quote No
Stock File
Stock Ref
Links
Stock Link
Quote Link
Quote File
Quote Ref
Cust no
Customer File
Cust Ref
Schedule
Staff No
Staff File
Staff Ref

StaffFile: The file that is the permanent store of all the staff details.
Expected number of records: 2
Access Type: Serial
Justification of access: For the staff file I have chosen serial access because of the size of the file. A serial access file would be suitable as when the search is done for the staff member loading the file will still be quick as there is only 2 members of staff. Random access is not necessary as the search time would not be increased a beneficial amount. Furthermore, I have chosen not to use sequential access since the data does not need to be ordered in any particular way as it would not affect the system or its use.

Objective	Outputs
Allow the user to add a new employee to the system, entering key information including: Full name, Address, Telephone number and national insurance number. This will then be stored in a permanent file.	N/A
Permit the user to change information stored on an employee e.g., Address and telephone number.	N/A
Ability to delete a member of staff that no longer works there.	N/A
Allow the user to search for staff via name.	Display of staff information including: Name, Home Address, Telephone number and National Insurance Number.
Include validation on all data entered when adding a new employee.	See error messages from validation design.
Ensure all information saved is backed up.	N/A

CustomerFile: The file that is the permanent store of all the customer details.
Expected number of records: 100-150
Access Type: Random access
Justification of access: For the customer file I have chosen random access because the file is much larger in size, and it would be beneficial to be able to quickly search for a customer's details for the staff when they are trying to produce a quote or invoice since it would make the overall process more efficient. Furthermore, I have not chosen serial access because the file size is much larger and so searching through the file would take much longer in a serial access file. However, I have not chosen sequential access because there was no need for the customers to be ordered in any particular way and would not benefit any users of the system.

Allow the user to enter a new customer to the system by storing: Full name, Address, Telephone number, and a basic description of the job in a permanent file.	N/A
Allow the user to delete a customer from the system when they are no longer using the company.	N/A
Have the ability to change the address or telephone number of a customer stored on the system.	N/A
Enable the user to search for customer using name or customer reference.	Display of the following customer information: Name, Home Address, Telephone number and a basic job description.
Ensure all data entered is validated.	See error messages from validation design.
Ensure all information saved is backed up.	N/A

QuoteFile: The file that is the permanent store of all the quote details.
Expected number of records: 150-200
Access Type: Serial
Justification of access: For the quotes file I have chosen serial access because there are multiple criteria that the quotes must be searched for such as quote reference and customer name. Therefore, it would be more useful to have all the data loaded at one time to easily retrieve the required information. I chose not to use random access since the need to search on multiple fields therefore this outweighs the fact that due to larger size of the file random access would have been more beneficial. Furthermore, I have not chosen sequential since there is no need for there to be an order such as date as it has no effect on the data retrieved and required by the staff.

Allow the user to add a new quote to the system by storing it in a permanent file. The information to be stored on a quote will be: Date the quote was produced, customer information, a job description, the number of days required to complete the job, a basic idea of what stock will be needed and an estimated price of the job.	N/A
Enable the user to delete a quote once the booking is finalised.	N/A
The ability to change the information on the quote including the estimated price and the number of days required.	N/A
Enable the user to search for a specific quote using quote reference or customer name.	Display the following information on a quote: Customer reference, basic stock required and the estimated price of the job.
Produce a calculation for the quote taking into account current stock prices, labour costs,	Display the cost of each of the following sections: Material costs, Labour costs, mileage and VAT.

number of days worked and mileage.	Also display the total cost at the end.
Validate all quote information upon entry.	See error messages from validation design.
Ensure all information saved is backed up.	N/A

InvoiceFile: The file that is the permanent store of all the invoice details.
Expected number of records: 100-150
Access Type: Serial
Justification of access: For the invoice file I have chosen serial access because the staff members will need to search for multiple criteria including through a name or to see which have been paid. This means that it would be more beneficial to have all the data loaded at one time unlike a random-access file. Random access would not be useful as the requirements for the stock file mean there are searches and sorts that suit a serial access file better. Even though the size of the file would suit a random access file the need for the searches and sorts its more important and useful for the system. Furthermore, I have not chosen sequential since there is no need for there to be an order such as date as it has no effect on the data retrieved and required by the staff.

Allow the user to add a new invoice to the system storing this in a permanent text file including the following information: Customer information, Date of the job, Description of the job.	N/A
Allow the user to be able to change information including date of the job and whether the job has been paid for.	N/A
The ability to delete an invoice from the system once it is no longer needed and the job has been paid.	N/A
Produce a multistage calculation for the invoice taking into account: number of days worked, mileage costs, current stock costs and labour costs.	Display the cost of each of the following sections: Material costs, Labour costs, mileage and VAT. Also display the total cost at the end.
Enable the user to be able to search for an invoice by name.	For each invoice searched display: Customer reference, price, and dates of the job.

Enable the user to be able to search for paid or unpaid invoices.	Display the customer reference and the status of the payment.
Ensure there is validation in place for all information entered excluding the description of the job.	See error messages from validation design.
Ensure all information saved is backed up.	N/A

StockFile: The file that is the permanent store of all the stock details.
Expected number of records: 20
Access Type: Serial
Justification of access: I have chosen serial access in this case because the staff will carry out searches and sorts on the information therefore, I believe that serial will be a better fit since it will mean all data is loaded at once making the processes easier to carry out. I have not chosen random access due to the fact that there are multiple searches and sorts which will be more efficient in serial but also because the file size is much smaller, so the quickness of random access is not as important. Furthermore, I have not chosen sequential since there is no need for there to be an order.

Allow the user to add additional stock to a permanent file including information such as: colour of paint, type of paint, volume of can, price and quantity.	N/A
Allow the user to delete stock from the system if it is no longer required or is no longer sold.	N/A
The ability to change the price and quantity of the stock.	N/A
Enable the user to search for stock using stock ID to display information.	Display the following stock information: Colour, price and quantity.
Enable the user to sort stock quantities from low to high.	Display the stock reference and the quantity of the stock in the sorted list.
Produce a message when stock quantities drop to a specific level.	Display a simple output message when stock drops so low.
Ensure information entered is validated.	See error messages from validation design.
Ensure all information saved is backed up.	N/A

LinksFile:
Expected number of records:
Access Type: Serial
Justification of access: I have chosen serial access as it means that all the data in the file will be loaded at once making it easier to link the files and data together. Furthermore, since the file will not be big the use of random access would be pointless, and efficiency would not increase a credible amount. The data stored in the links file would not need to be in any particular order so sequential access would be unnecessary.

Variable Tables

Staff:

Field Name	Description	Type	Length	Sample
staffref	Unique ID for each member of staff. Key Field.	Int	Short	2
fnamestaff	Staff members first name.	Char	15	Stuart
lnamestaff	Staff members last name.	Char	15	Watson
oneadstaff	1st line of a staff members home address.	Char	30	15 Cherry Tree Close
twoadstaff	2nd line of a staff members home address.	Char	15	Kiveton Park
threeadstaff	3rd line of a staff members home address.	Char	15	Sheffield
pcodestaff	The postcode for the staff members home address	Char	8	S26 5QQ
telnostaff	The contact number for the member of staff.	Char	11	07743117706
ninum	The national insurance number for the member of staff.	Char	9	QQ123456A
Username	Unique username to allow a staff member to log on.	Char	20	TSmith
Password	Unique password to allow a staff member to log on.	Char	15	TylerSm!th49
Loa	Hierarchical access to allow staff member specific access to areas of the system.	Int	short	3

Customers:

Field Name	Description	Type	Length	Sample
custref	Unique ID for each customer. Key Field.	Int	Short	5
fnamecust	Customers first name.	Char	15	Emily
lnamecust	Customers last name.	Char	15	Carlton
oneadcust	1st line of a customer's home address.	Char	30	26 Wiltshire Bank
twoadcust	2nd line of a customer's home address.	Char	15	Sothall
threeadcust	3rd line of a staff members home address.	Char	15	Sheffield
pcodecust	The postcode for the customers home address	Char	8	S20 2NT
telnocust	The contact number for the customer.	Char	11	07795236573

Quotes:

Field Name	Description	Type	Length	Sample
quoteref	Unique ID for each quote. Key Field.	Int	Short	3
custno	Unique ID for the customer relating to the job. Foreign Key.	Int	Short	5
mainjobdesc	An abbreviated description of the job requirements.	Char	50	Bedroom Strip WP Paint ceil and walls
numofdays	To produce the quote calculation the estimated number of days	Int	Short	7

	required to complete the job is used.			
mileage	The travel costs for commuting to the job each day.	Int	Short	60
totalcost	The final price of the job all added together.	Int	Short	590

Stock:

Field Name	Description	Type	Length	Sample
stockref	Unique ID for each item of stock. Key field.	Int	Short	3
quantity	The amount of each item of stock currently stored at that time.	Int	Short	5
colour	The colour of the paint.	Char	15	Satin White
volume	The volume of the tin of paint in litres.	Int	Short	50
type	The category of paint used.	Char	10	Matte
stockprice	The cost of the item of stock	Float	2.2	95.99

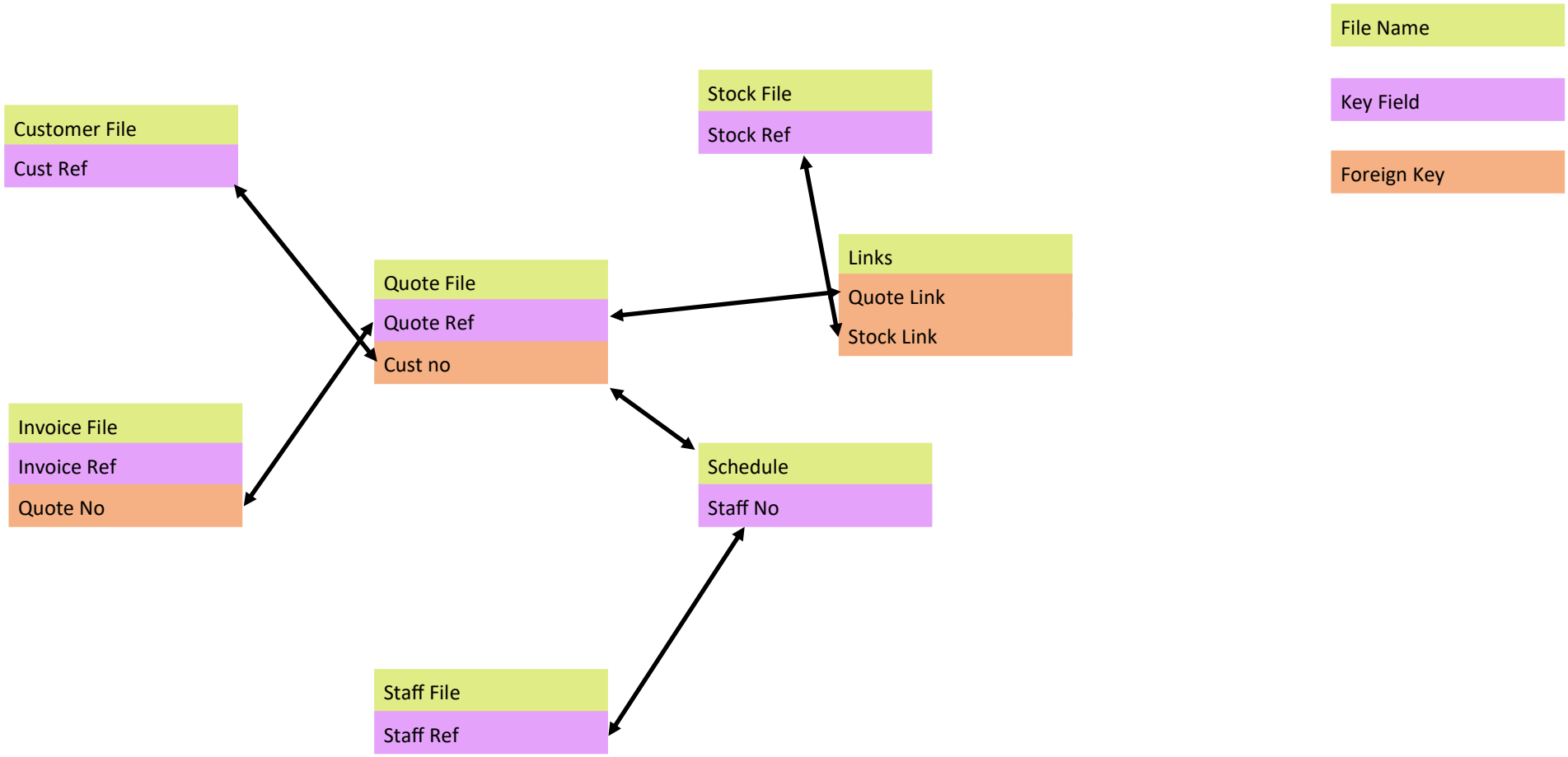
Invoices:

Field Name	Description	Type	Length	Sample
invoiceref	Unique ID for each invoice. Key field.	Int	Short	3
custnum	The unqiue identifier of customer. Foreign key	Int	Short	3
jobstartdate		Char	10	20/06/2023

	The date that work on the job began.			
jobenddate	The final date of the job.	Char	10	27/06/2023
finaldesc	Abbreviated description of the job after it has been completed.	Char	50	Bedroom Strip WP Paint ceil and walls
inumofdays	The number of days that the job took from the start date to the final date.	Int	short	7
matprices	The price of the materials paid for during the duration of the job.	Float	3.2	100.00
labourprices	The day to day price of the job.	Int	Short	480
mileage	Cost of fuel for travelling.	Int	short	60
totalprice	The final price of the job all added together.	Float	4.2	0590.35
paid	Identifies whether or not the invoice has been paid or not.	Char	1	Y

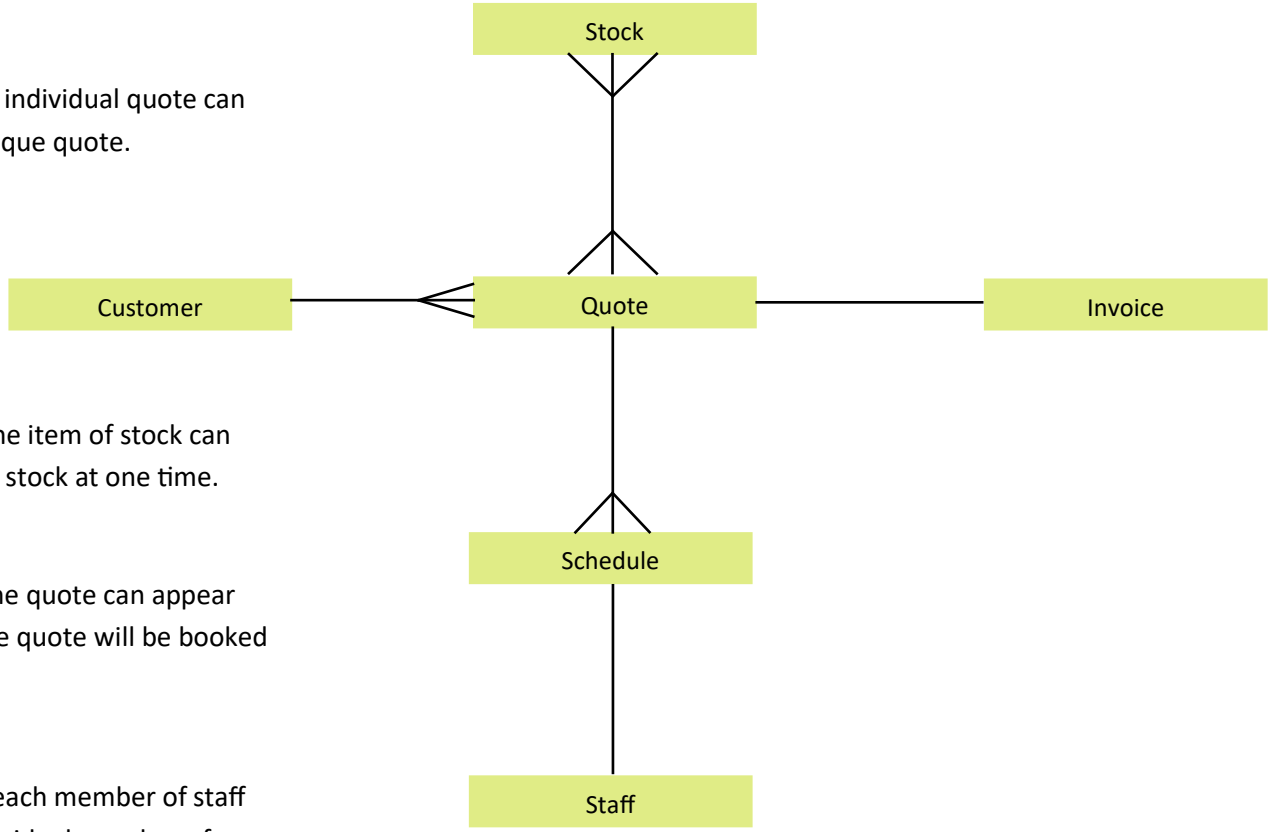
Links File:

Field Name	Description	Type	Length	Sample
linksquoteref	The unique reference of the quote that is being linked to the stock.	Int	Short	3
linksquantity	The quantity of each item of stock need for the quote	Int	Short	3
linksstockref	The unique reference of the stock that is being linked to the quote.	int	Short	3


File Name
Key Field
Foreign Key
Customer File
Cust Ref
Stock File
Stock Ref
Links
Quote Link
Stock Link
Quote File
Quote Ref
Cust no
Invoice File
Invoice Ref
Quote No
Schedule
Staff No
Staff File
Staff Ref

The relationship between a customer and a quote is 1-many since one customer can have multiple quotes however an individual quote can only belong to one customer.

The relationship between a quote and an invoice is 1-1 since one individual quote can only have one invoice and one invoice can only belong to one unique quote.



The relationship between stock and quote is many-many since one item of stock can belong to many quotes and one quote will include many items of stock at one time.

The relationship between a quote and a schedule is 1-many as one quote can appear multiple times on a schedule as it is done by blocks of time so one quote will be booked over multiple blocks of times.

The relationship between the schedule and the staff is 1-1 since each member of staff has their own schedule so a schedule will only be viewed per individual member of staff.

Validation

Staff:

Field Name	Description	Type	Length	Sample	Validation Type	Further description	Expected Output
staffref	Unique ID for each member of staff. Key Field.	Int	Short	2	File Check	Read back the file to check if the staffref entered is new or whether it already exists within the file as all staff references must be unique on the system, cannot be duplicates.	This staff reference already exists. Please re-enter a different number.
fnamestaff	Staff members first name.	Char	15	Stuart	Presence	Makes sure that data has been entered to record to the file.	Ensure that you have entered a first name.
lnamestaff	Staff members last name.	Char	15	Watson	Presence	Makes sure that data has been entered to record to the file.	Ensure that you have entered a last name.
oneadstaff	1st line of a staff members home address.	Char	30	15 Cherry Tree Close	Presence	Makes sure that data has been entered to record to the file.	Please check to see if line 1 of the address has been entered.
twoadstaff	2nd line of a staff members home address.	Char	15	Kiveton Park	Presence	Makes sure that data has been entered to record to the file.	Please check to see if line 2 of the address has been entered. If it is not applicable please

							use ‘N/A’.
threeadstaff	3rd line of a staff members home address.	Char	15	Sheffield	Presence	Makes sure that data has been entered to record to the file.	Please check to see if line 3 of the address has been entered.
pcodestaff	The postcode for the staff members home address	Char	8	S26 5QQ	Format	AA9A 9AA A9A 9AA A9 9AA A99 9AA AA9 9AA AA99 9AA Validates the postcode for all UK formats checking the length and that all letters and digits are in the correct space for the corresponding length.	Please check that the letters are in the correct place. Please check that the numbers are in the correct place. Please make sure that the postcode is of the correct length including spaces.
telnostaff	The contact number for the member of staff.	Char	11	07743117706	Format	07NNNNNNNN Validates a UK mobile number ensuring it starts with a zero and is exactly 11 digits long and are all digits.	Please make sure that the mobile number entered begins with 07. Please make sure that the mobile number entered is exactly

							11 digits long.
ninum	The national insurance number for the member of staff.	Char	9	QQ 12 34 56 A	Format	LL NNNNNN L Validates a national insurance number ensuring it is of the correct length, with all spaces, digits and letters in the correct place.	Please make sure that the NI number is 13 characters in length including spaces. Please make sure that there are the correct spaces within the NI number. Please make sure that the letters are typed correctly an in the right place. Please make sure that the numbers are typed correctly an in the right place.

username	The staff members unique login for the system	char	20	TSmith	Unique File Check	Read back the file to check if the username entered is new or whether it already exists within the file.	This username is already in use. Pleas try again with a different username.
password	Staff members unique password for logging onto the system.	Char	15	TylerSm!th41	Password criteria check	Validates that the new password entered is of at least 8 characters and has either a number or piece of punctuation within it.	Please make sure that the password is at least 8 characters long. Please make sure that the password has either a piece of punctuation or a number.
Loa	Hierarchical access for the staff member to allow them to do certain administration roles within the company.	int	1	3	Range	Makes sure that the level of access is between 1-3 as these are the levels I have chosen. 3 being the most access.	Please make sure the level of access is between 1-3.

						1 being the least.	

Customers:

Field Name	Description	Type	Length	Sample	Validation Type	Further Description	Expected Output
custref	Unique ID for each customer. Key Field.	Int	Short	5	File Check	Read back the file to check if the custref entered is new or whether it already exists within the file as all customer references must be unique on the system, cannot be duplicates.	This customer reference already exists. Please re-enter a different number.
fnamecust	Customers first name.	Char	15	Emily	Presence	Makes sure that data has been entered to record to the file.	Ensure that you have entered a first name.
lnamecust	Customers last name.	Char	15	Carlton	Presence	Makes sure that data has been entered to record to the file.	Ensure that you have entered a last name.
1adcust	1st line of a customer's home address.	Char	30	26 Wiltshire Bank	Presence	Makes sure that data has been entered to record to the file.	Please check to see if line 1 of the address has been entered.
2adcust	2nd line of a customer's home address.	Char	15	Sothall	Presence	Makes sure that data has been entered to record to the file.	Please check to see if line 2 of the address has been entered.

							If it is not applicable please use 'N/A'.
3adcust	3rd line of a staff members home address.	Char	15	Sheffield	Presence	Makes sure that data has been entered to record to the file.	Please check to see if line 3 of the address has been entered.
pcodecust	The postcode for the customers home address	Char	8	S20 2NT	Format	AA9A 9AA A9A 9AA A9 9AA A99 9AA AA9 9AA AA99 9AA Validates the postcode for all UK formats checking the length and that all letters and digits are in the correct space for the corresponding length.	Please check that the letters are in the correct place. Please check that the numbers are in the correct place. Please make sure that the postcode is of the correct length including spaces.
telnocust	The contact number for the customer.	Char	11	07795236573	Format	0NNNNNNNNN Validates a UK mobile number ensuring it starts with a zero and is exactly 11 digits long and are all digits.	Please make sure that the mobile number entered begins with a 0. Please make sure that the mobile

							number entered is exactly 11 digits long.

Quotes:

Field Name	Description	Type	Length	Sample	Validation Type	Further Description	Expected Output
quoteref	Unique ID for each quote. Key Field.	Int	Short	3	File Check	Read back the file to check if the quoteref entered is new or whether it already exists within the file as all quote references must be unique on the system, cannot be duplicates.	This quote reference already exists. Please re-enter a different number.
custno	Unique ID for the customer relating to the job. Foreign Key.	Int	Short	5	File Check	Read back the file to check if the custno entered is new or whether it already exists within the file.	This customer number already exists. Please re-enter a different number.
mainjobdesc	An abbreviated description of the job requirements.	Char	50	Bedroom Strip WP Paint ceil and walls	Presence	Makes sure that data has been entered to record to the file.	Please ensure you have entered a description of the job for the quote.

eststock	An estimated idea of what stock will be required for the job.	Int	Short	1 3 5	Presence	Makes sure that data has been entered to record to the file.	Please ensure you have entered the stock ID's for the stock you think you may require.
qnumofdays	To produce the quote calculation the estimated number of days required to complete the job is used.	Int	Short	7	Range	Makes sure that the data entered is appropriate in context to the variable – ensure user does not enter 70 days instead of 7 etc.	Please ensure you have entered the correct number of days you expect to work.
matprices	The price of the materials required for the job. This will be linked automatically from the stock file so will have already been validated upon entry of the item of stock.	Int	Short	100			
Mileage	Cost of fuel for travelling to a job each day.	Int	Short	60	Presence	Makes sure that data has been entered to allow a calculation for the job to happen.	Please ensure you have entered the milage costs.
totalprice	The final price of the job all added together.	Int	Short	710			

	This will be outputted to the screen not entered so cannot be validated upon entry.						

Stock:

Field Name	Description	Type	Length	Sample	Validation Type	Further Description	Expected Output
stockref	Unique ID for each item of stock. Key field.	Int	Short	3	File Check	Read back the file to check if the stockref entered is new or whether it already exists within the file as all stock references must be unique on the system, cannot be duplicates.	This stock reference already exists. Please re-enter a different number.
quantity	The amount of each item of stock currently stored at that time.	Int	Short	5	Range check	Makes sure that the data entered is appropriate in context to the variable – eg quantity entered is 10 and not 100	Please ensure you have entered the quantity of this item you currently have correctly.
colour	The colour of the paint.	Char	15	Satin White	Presence	Makes sure that data has been entered to record to the file.	Please ensure that the colour of the paint has been

							entered. If colour isn't applicable please use N/A
volume	The volume of the tin of paint in litres.	Int	Short	50	Range	Makes sure that the data entered is appropriate in context to the variable – eg volume entered is 10l not 100l	Please ensure that the volume of the paint entered is correct.
type	The category of paint used.	Char	10	Matte	Presence	Makes sure that data has been entered to record to the file.	Please ensure that the type of the paint has been entered. If type of paint isn't applicable please use N/A
stockprice	The cost of the item of stock	Int	short	96	Range	Ensure the price of the stock entered is a suitable value and has not been mistyped.	Please ensure the price entered is correct.

Invoices:

Field Name	Description	Type	Length	Sample	Validation Type	Further Description	Expected Output
invoiceref	Unique ID for each	Int	Short	3	File Check	Read back the file to check if the	This invoice reference

	invoice. Key field.					invoiceref entered is new or whether it already exists within the file as all invoice references must be unique on the system, cannot be duplicates.	already exists. Please re-enter a different number.
custno	The unqiue identifier of customer. Foreign key	Int	Short	3	File Check	Read back the file to check if the custno entered is new or whether it already exists within the file as all customer references must be unique on the system, cannot be duplicates.	This customer number already exists. Please re-enter a different number.
jobstartdate	The date that work on the job began.	Char	10	20/06/2023	Format DD/MM/YYYY Range check – using systems clock	Ensures that the date entered follows a particular format as represented. Checking that the dates are correct for the corresponding month. That the forward slashes are in the correct place.	Please ensure that the dates are between the 1st and 31st for this month. Please ensure the dates are between the 1st and 30th for this month.

						Checks the months are in the range 01-12. Also checks that the date entered is in the future and won't allow a date from the past to be used by comparing to the systems clock.	Please ensure the dates are between the 1st and 28th for this month. Please ensure the dates are between the 1st and 29th for this month. Please check the month is in the range 01-12. Please check that the forward slashes are in the correct place. Please check that date entered is after the current date.
jobenddate	The final date of the job.	Char	10	27/06/2023	Format DD/MM/YYYY Range check	Ensures that the date entered follows a particular format as represented. Checking	Please ensure that the dates are between the 1st and 31st

						that the dates are correct for the corresponding month. That the forward slashes are in the correct place. Checks the months are in the range 01-12. Checks that the final date entered is later than the start date entered	for this month. Please ensure the dates are between the 1st and 30th for this month. Please ensure the dates are between the 1st and 28th for this month. Please ensure the dates are between the 1st and 29th for this month. Please check the month is in the range 01-12. Please check that the forward slashes are in the correct place. Please check that

							the final day is after the start date.
finaldesc	Abbreviated description of the job after it has been completed.	Char	50	Bedroom Strip WP Paint ceil and walls	Presence	Makes sure that data has been entered to record to the file.	Please ensure you have entered a description of the job for the invoice.
inumofdays	The number of days that the job took from the start date to the final date.	Int	Short	7	Presence	Makes sure that data has been entered to allow a calculation for the job to happen.	Please ensure you have entered the number of days you worked.
matprices	The price of the materials paid for during the duration of the job.	Int	Short	100	Range	Makes sure that data has been entered to allow a calculation for the job to happen.	Please ensure you have entered the price of materials.
Mileage	Cost of fuel for travelling to a job each day.	Int	Short	60	Presence	Makes sure that data has been entered to allow a calculation for the job to happen.	Please ensure you have entered the milage costs.
totalprice	The final price of the job all added together. This will be calculated within the program and is not entered by the user and	Float	4.2	0590.35			

	therefore cannot be validated upon entry.						
paid	Identifies whether or not the invoice has been paid or not.	Int	short	1	Range	Makes sure that data is between 0 and 1. 0 = No 1 = Yes	Please ensure you have entered whether or not the job has been paid for by choosing either Y or N.

Representation of Multi-dimensional Array

The Array will use sequential access as the data stored needs to be ordered by the date and times of a booking and this cannot be done is serial or random access therefore they could not be used.

Each staff member will be represented by a number. Currently there are two members of staff so 1-2.

Each day of the week will be represented by a number.
In this case, January 1st will be 1, January 2nd will be 2, May 3rd will be 124 (31 (Jan) + 29 (Feb) + 31 (Mar) + 30 (Apr) + 3 (May).

Each time slot will be represented by a number. The working hours are between 7am – 7pm and therefore will be numbered 7-19.

	7	8	9	10	11	12	13	14	15	16	17	18	19
1	4	4	4	4	4								
2													
3													
4													
5		7	7	7	7	7	0	7	7	7			
6													
7													
364													
365													
366													

This booking schedule shows the upcoming work for staff member 2.

From the schedule you can see he is set to work from 7am – 11am on 1st January and 8am – 4pm on 5th January.

I will use the quote references to store bookings on the schedule for example, on the 5th January staff member 2 will be completing the quote referenced 2.

The '0' on the schedule demonstrates the lunch break for the worker as 0 will not be used as a quote reference they will start from 1.

I will use the 3D Array named BookingSchedule.

The field will be bookingsch (date, hour, staffno)

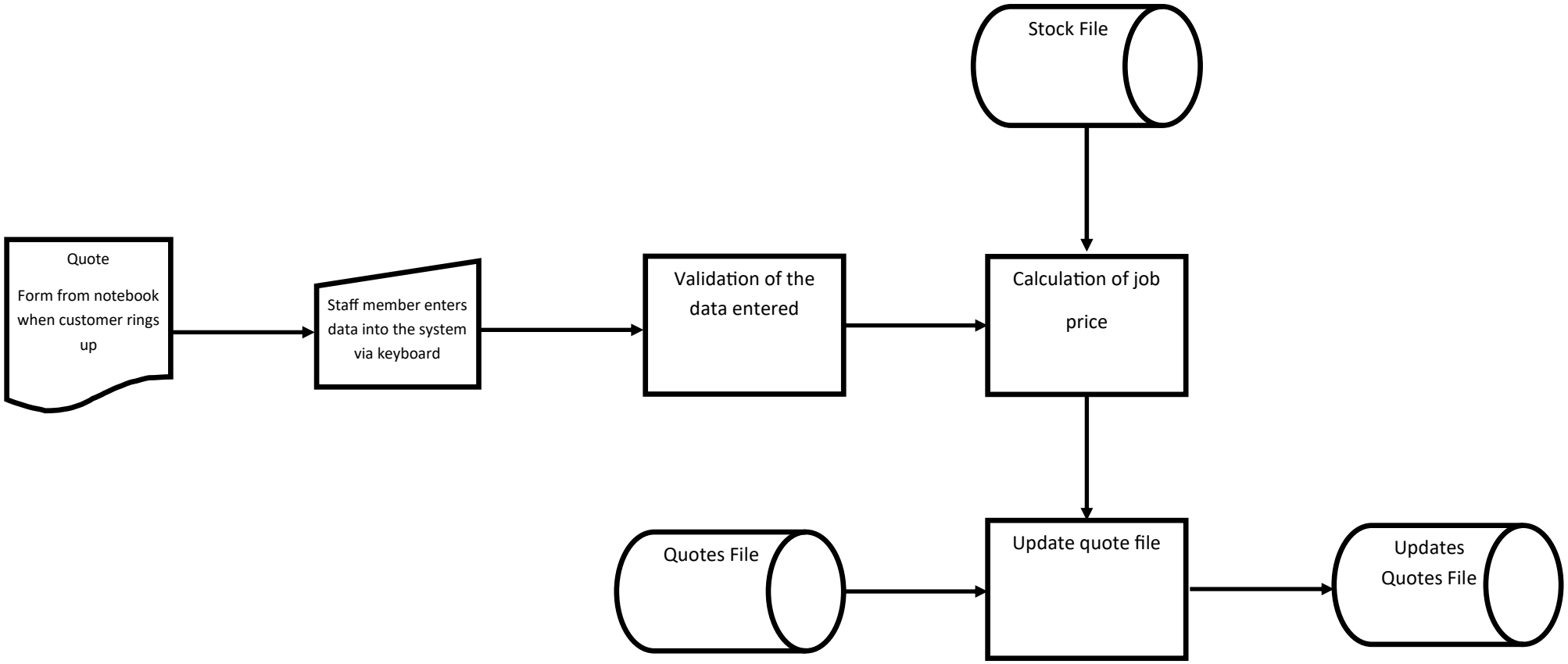
The example shows:

bookingsch (1, 7, 2) = "4"
bookingsch (1, 8, 2) = "4"
bookingsch (1, 9, 2) = "4"
bookingsch (1, 10 ,2) = "4"
bookingsch (1, 11, 2) = "4"

bookingsch (5,8, 2) = "7"
bookingsch (5, 9, 2) = "7"
bookingsch (5,10 , 2) = "7"
bookingsch (5, 11, 2) = "7"
bookingsch (5,12 , 2) = "7"
bookingsch (5, 13, 2) = "0"
bookingsch (5,14 , 2) = "7"
bookingsch (5, 15, 2) = "7"
bookingsch (5,16 , 2) = "7"

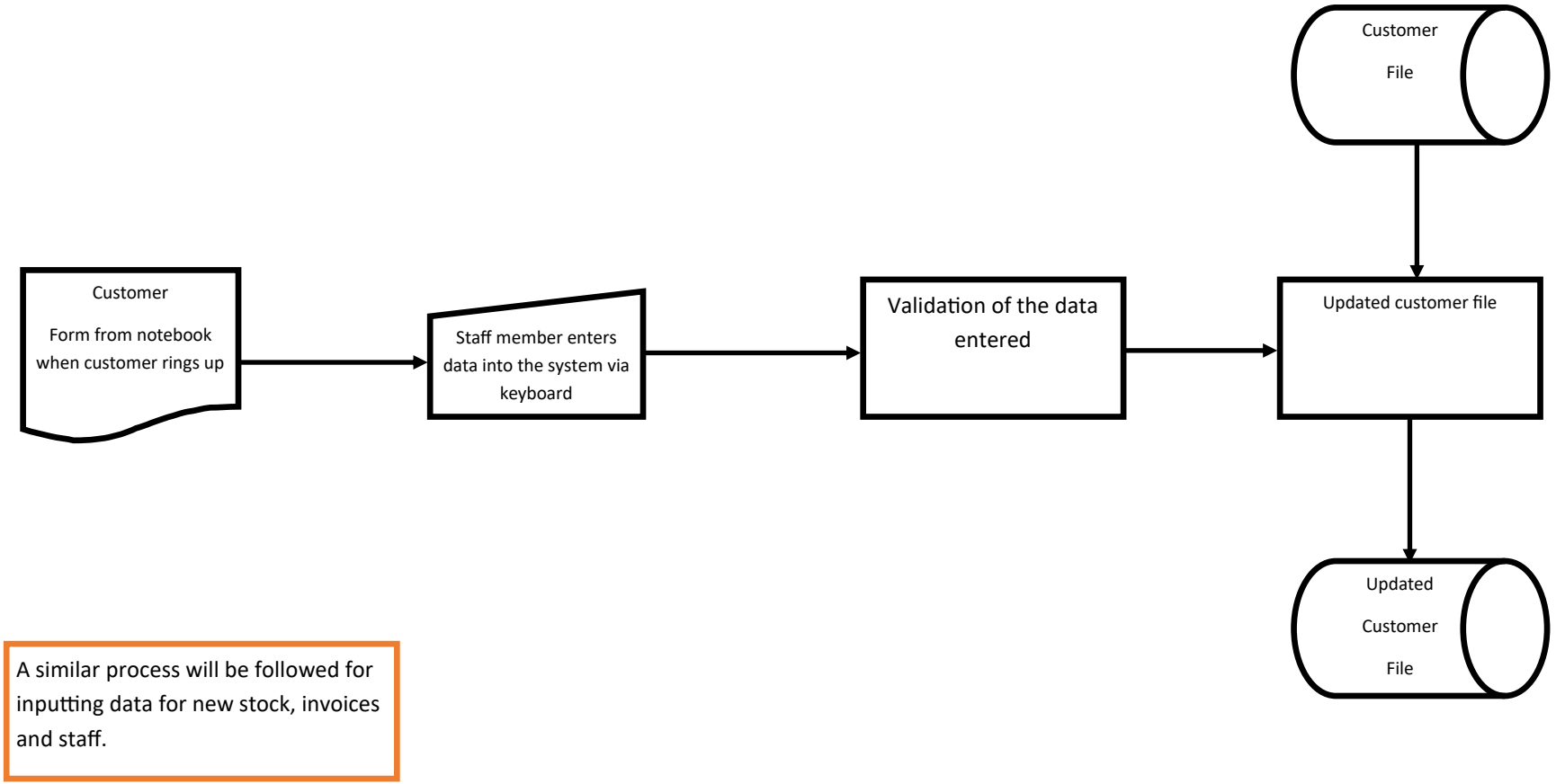
Relationships between the data and the processes

Adding a quote:



Relationships between the data and the processes

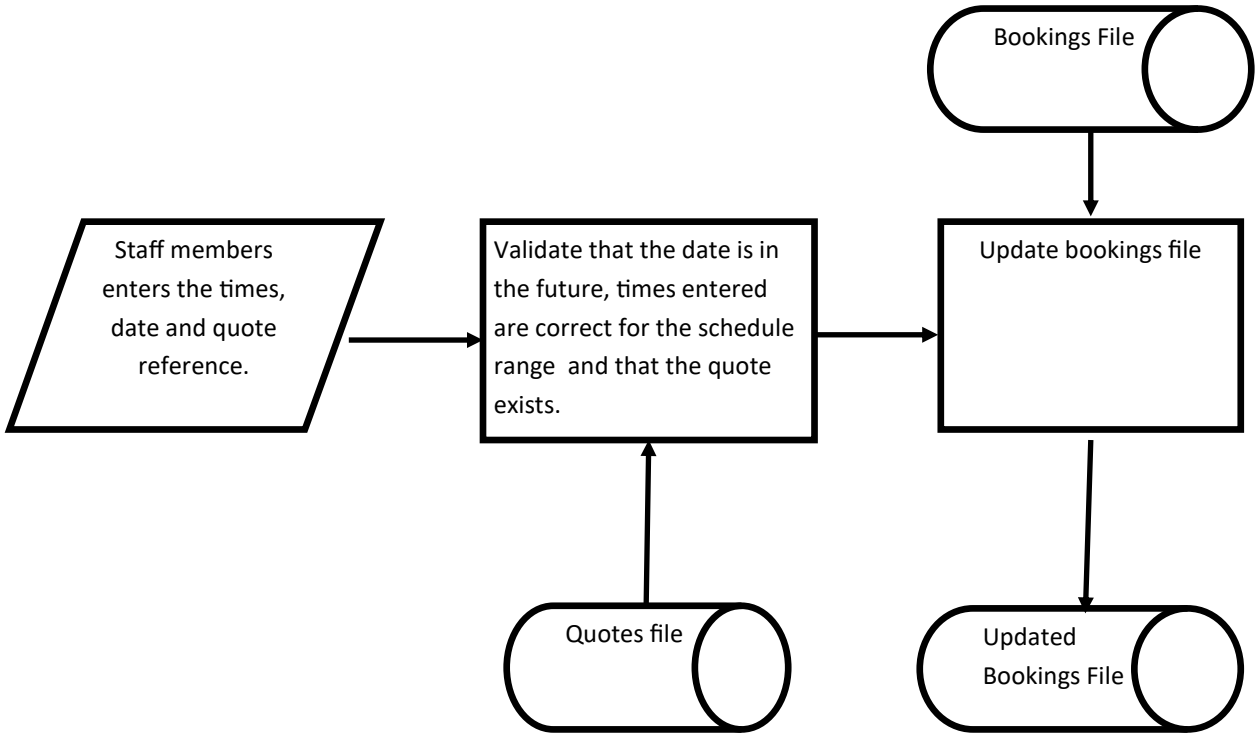
Adding a customer:



A similar process will be followed for inputting data for new stock, invoices and staff.

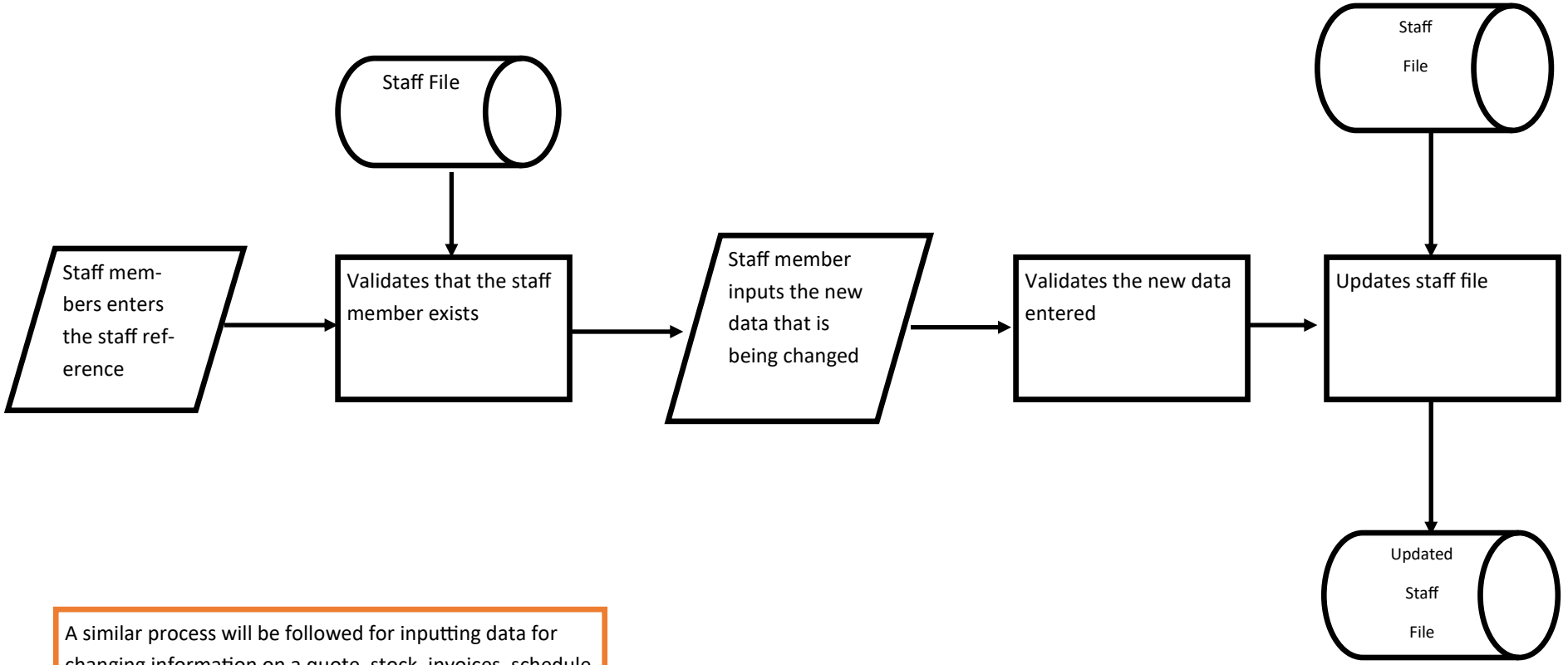
Relationships between the data and the processes

Adding a booking to the schedule:



Relationships between the data and the processes

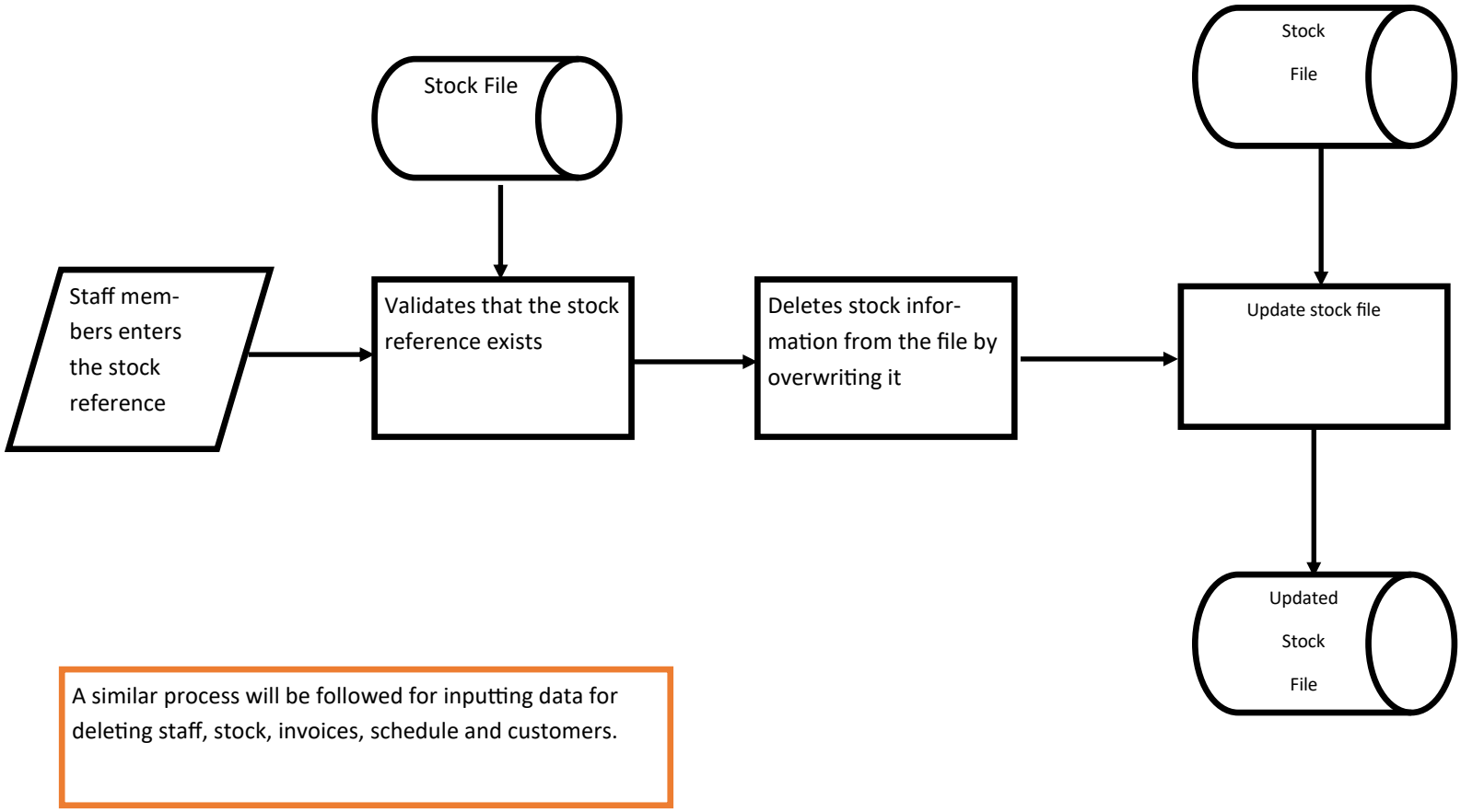
Changing staff information:



A similar process will be followed for inputting data for changing information on a quote, stock, invoices, schedule and customers.

Relationships between the data and the processes

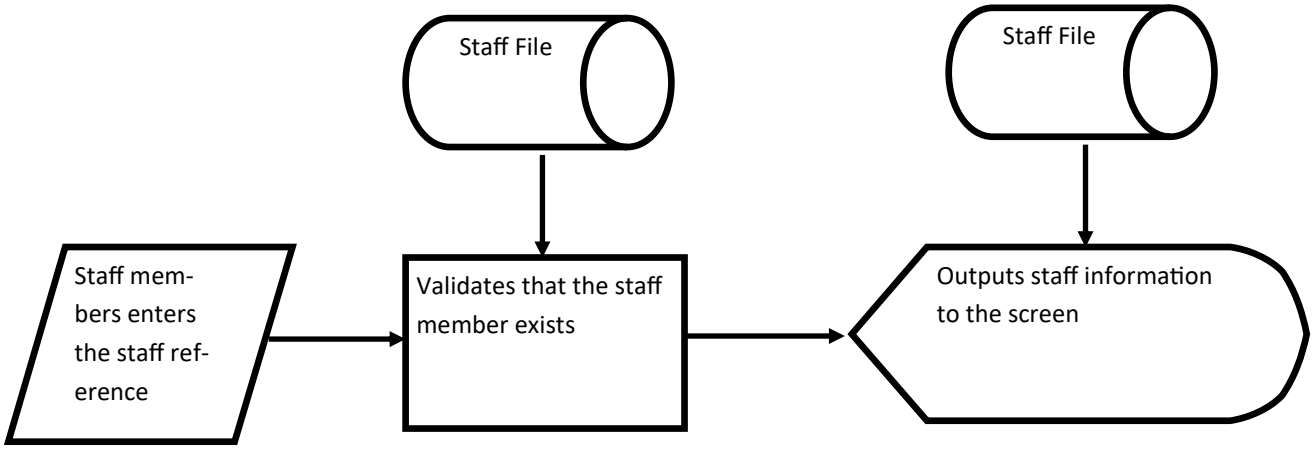
Deleting stock:



A similar process will be followed for inputting data for deleting staff, stock, invoices, schedule and customers.

Relationships between the data and the processes

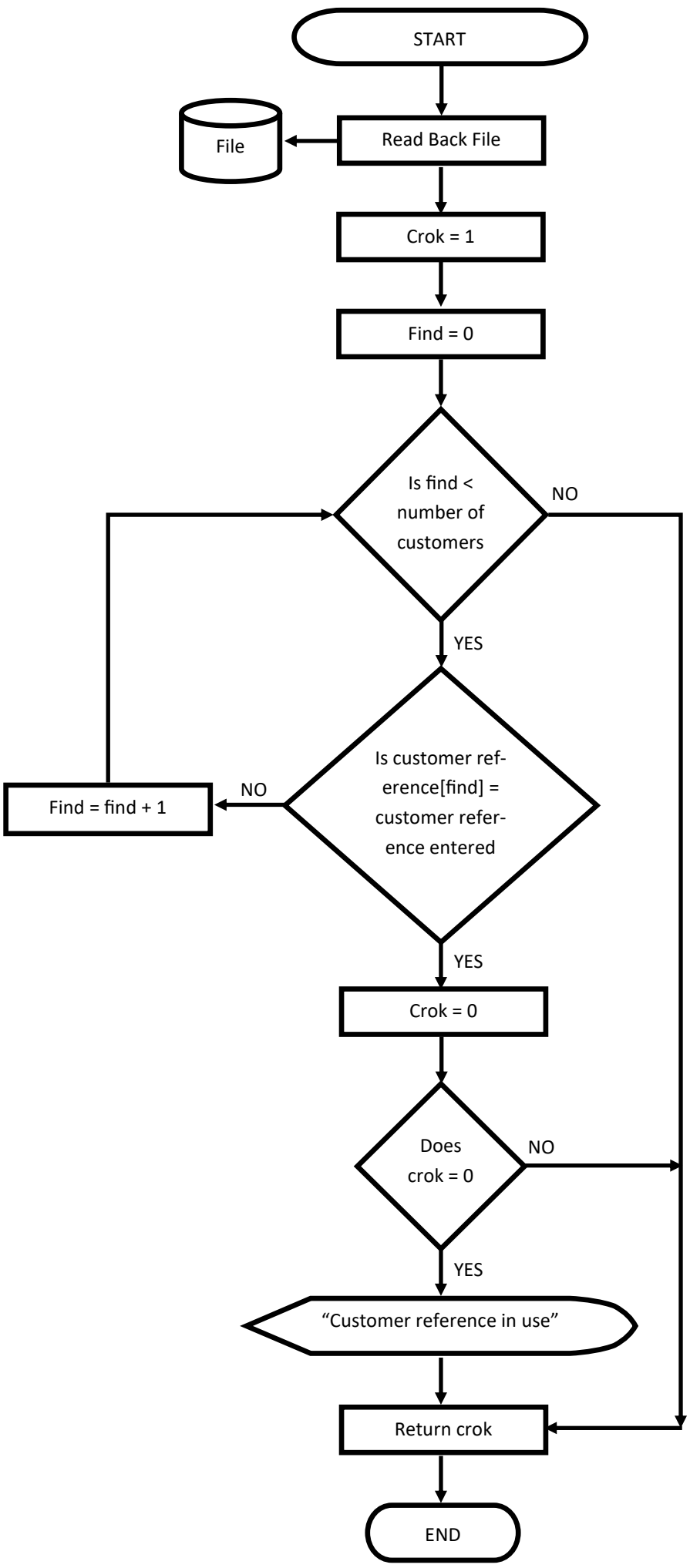
Viewing a staff member:



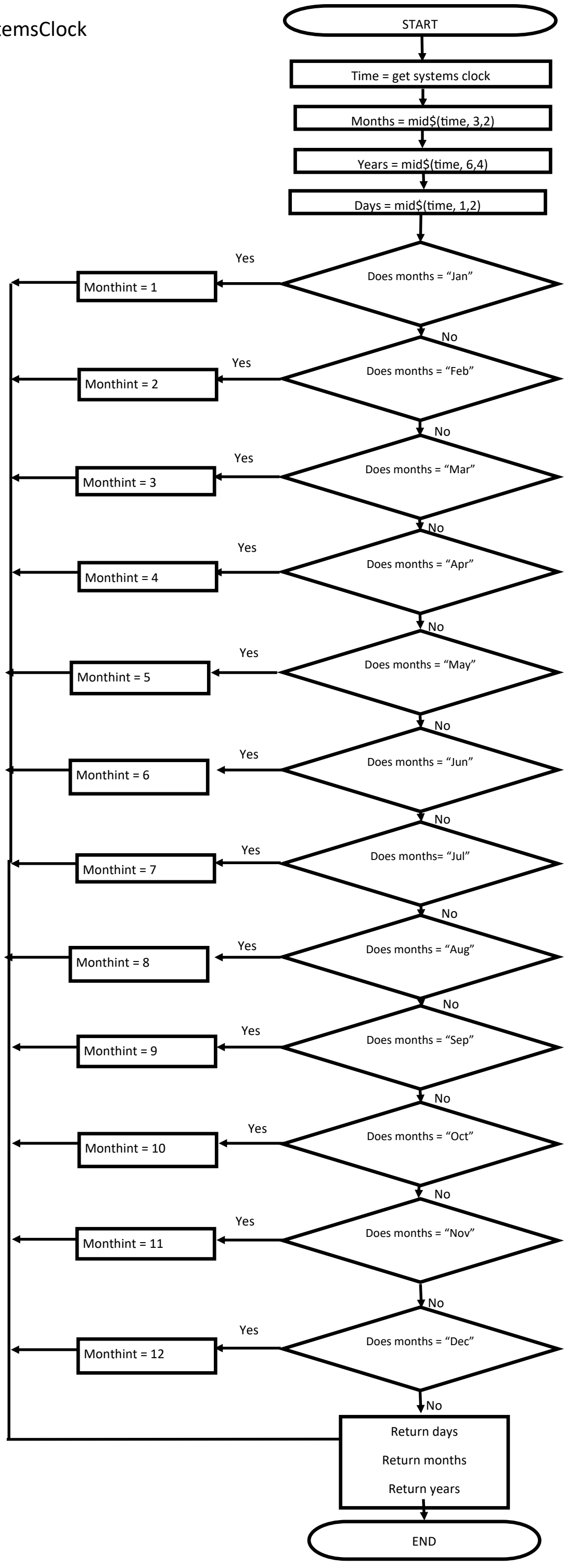
A similar process will be followed for inputting data for deleting quotes, stock, invoices, schedule and customers.

Proc UniqueVal

A similar program / flowchart will be used for UsernameVal and PassVal.



Proc SystemsClock



Serial File – Stock

ReadBackStockFile:

```
Proc ReadBackStockFile
Open StockFile
Read nsti
For count = 1 to nsti
        Read stockref(count)
        Read quantity(count)
        Read colour(count)
        Read volume(count)
        Read type(count)
        Read stockprice(count)
End for
Close StockFile
End proc
```

ReWriteStockFile:

```
Proc ReWriteStockFile
Open StockFile
Write nsti
For count = 1 to nsti
        Write stockref(count)
        Write quantity(count)
        Write colour(count)
        Write volume(count)
        Write type(count)
        Write stockprice(count)
End for
Close StockFile
```

End proc

Add Stock:

```
Proc AddStock
unqiue = 0, qrange = 0, cpres = 0, vrange = 0, tpres= 0, prange = 0
Do Proc ReadBackStockFile
While unique = 0
        output “Enter new stock ID: “
        input stockref[nsti]
        unique = Do Proc UniqueVal(stockref[nsti])
Endwhile
While cpres = 0
        output “Enter colour of item: “
        input colour[nsti]
        cpres = Do Proc PresenceCk(colour[nsti)
Endwhile
While tpres = 0
        output “Enter type of paint: “
        input type[nsti]
        tpres = Do Proc PresenceCk(type[nsti])
Endwhile
While prange = 0
        output “Enter price of stock:
        input stockprice[nsti]
        prange = Do Proc RangeCk(stockprice[nsti])
Endwhile
While vrange = 0
        output “Enter volume of item: “
        input volume[nsti]
        vrange = Do Proc RangeCk(volume[nsti])
```

```
Endwhile
While qrage = 0
        output “Enter current quantity of stock: “
        input quantity[nsti]
        qrange = Do Proc RangeCk(quantity[nsti]
Endwhile
nsti = nsti + 1
Do Proc ReWriteStockFile
End proc
```

Delete Stock:

```
Proc DeleteStock
Do Proc ReadBackStockFile
Output “Enter stock ID: “
Input stockref[nsti]
For find = 1 to nsti
        If(find[nsti] = stockref)
                For del = find to nsti
                        stockref[del] = stockref[del+1]
                        quantity[del] = quantity[del+1]
                        colour[del] = colour[del+1]
                        volume[del] = volume[del+1]
                        type[del] = type[del +1]
                        stockprice[del] = stockprice[del+1]
                End for
        End if
End for
nsti = nsti –1
Do Proc ReWriteStockFile
```

```
End proc
```

Change Stock Price:

```
Proc ChangeStockP
prange = 0
Do Proc ReadBackStockFile
output “Enter stock ID: “
input stockref[nsti]
For find = 1 to nsti
        If(find[nsti] = stockref)
                while prange = 0
                        output “Enter new price: “
                        input stockprice[nsti]
                        prange = Do Proc RangeCk(stockprice[nsti])
                end while
        End if
        Else
                output  “Stock ID does not exist”
End for
Do Proc ReWriteStockFile
End proc
```

Change Stock Quantity:

```
Proc ChangeStockQ
qrange = 0
Do Proc ReadBackStockFile
output “Enter stock ID: “
input stockref[nsti]
```

```
For find = 1 to nsti
        If(find[nsti] = stockref)
                while qrange= 0
                        output “Enter current quantity: “
                        input quantity[nsti]
                        prange = Do Proc RangeCk(quantity[nsti])
                end while
        End if
        Else
                Output “Stock ID does not exist”
End for
Do Proc ReWriteStockFile
End proc
```

View Stock by stock reference:

```
Proc ViewStockRef
output “Enter stock ID: “
input stockref[nsti]
Do Proc ReadBackStockFile
For find = 1 to nsti
        If(find[nsti] = stockref)
                output “Colour: “, colour[nsti]
                output “Price: “ , stockprice[nsti]
                output “Quantity: “ , quantity[nsti]
        End if
        Else
        output “ Stock ID does not exist.”
End for
End proc
```

Sort Stock Quantities:

```
Proc SortStockQ
Do Proc ReadBackStockFile
Flag = false
For find= 1 to quantity[nsti]
        If find > quantity[nsti+1]
                temp = quantity[find]
                quantity[find] = quantity[find+1]
                quantity[find+1] = temp
                flag = true
        End if
End for
nsti = nsti –1
End proc
```

Find low stock:

```
Proc FindLowStock
Do Proc ReadBackStockFile
For find = 1 to nsti
        If ( quantity[find]= 0)
        output “Stock ID “,stockref(find)
        End if
End for
End proc
```

Serial File – Staff

ReadBackStaffFile:

```
Proc ReadBackStaffFile
Open StaffFile
Read nsi
For count = 1 to nsi
        read staffref (count)
        read fnamestaff (count)
        read lnamestaff (count)
        read oneadstaff (count)
        read twoadstaff (count)
read threeadstaff(count)
        read pcodestaff (count)
        read telnostaff (count)
        read ninum (count)
        read username(count)
        read password(count)
        read loa(count)
End for
Close StaffFile
End proc
```

ReWriteStaffFile:

```
Proc ReWriteStaffFile
Open StaffFile
Write nsi
For count = 1 to nsi
```

write staffref (count)

write fnamestaff (count)

write lnamestaff (count)

write oneadstaff (count)

write twoadstaff (count)

write threeadstaff(count)

write pcodestaff (count)

write telnostaff (count)

write ninum (count)

write username(count)

write password(count)

write loa(count)

End for

Close StaffFile

End proc

<u>Add Staff:</u>

Proc AddStaff

unique = 0, presf = 0, presl = 0, presaone = 0, presatwo =0, presathree =0, pcode = 0, telnum=0, nino =0 , uniqueuser=0, uniquepass=0, lev=0

Do Proc ReadBackStaffFile

While unique = 0

output “Enter new staff reference: ”

input staffref [nsi]

unique = Do Proc UniqueVal(staffref [nsi])

Endwhile

```
While presf = 0
        output “Enter first name: “
        input fnamestaff [nsi]
        presf = Do Proc PresenceCk(fnamestaff [nsi])
Endwhile
While presl = 0
        output “Enter last name: “
        input lnamestaff [nsi]
        presl = Do Proc PresenceCk(lnamestaff [nsi])
Endwhile
While presaone = 0
        output “Enter address line 1: “
        input oneadstaff [nsi]
        presa1 = Do Proc PresenceCk(oneadstaff [nsi])
Endwhile
While presatwo = 0
        output “Enter address line 2 “
        input twoadstaff [nsi]
        presatwo = Do Proc PresenceCk(twoadstaff [nsi])
Endwhile
While presathree = 0
        output “Enter address line 3: “
        input 3adstaff [nsi]
        presathree = Do Proc PresenceCk(threeadstaff [nsi])
Endwhile
While pcode = 0
        output “Enter postcode: “
        input pcodestaff [nsi]
        pcode = Do Proc PostcodeVal(pcodestaff [nsi])
Endwhile
While telnum = 0
```

```
        output “Enter mobile number: “
        input telnostaff [nsi]
        telnum = Do Proc TelFormatCk(telnostaff [nsi])
Endwhile
While nino = 0
        output “Enter national insurance number: “
        input ninum [nsi]
        nino = Do Proc NIVal(ninum [nsi])
Endwhile
While uniqueuser=0
        output “Enter username: “
        input username [nsi]
        username = Do Proc UniqueVal(username [nsi])
End while
While unqiuepass =0
        output “ Enter password (At least 8 characters with at least one piece of punctuation or
        number) : ”
        input password [nsi]
        password = Do Proc PasswordCk(password [nsi])
End while
While lev = 0
        output “Enter level of access: “
        input loa [nsi]
        lev = Do Proc RangeCk(loa [nsi])
End while
nsi = nsi + 1
Do Proc ReWriteStaffFile
End proc
```

<u>Delete Staff:</u>

```
Proc DeleteStaff
Do Proc ReadBackStaffFile
output “Enter the staff reference: “
input staffref[nsi]
for find = 1 to nsi
        if (find[nsi] = staffref)
                for del = find to nsi
                        staffref[del] = staffref[del+1]
                        fnamestaff[del] = fnamestaff[del+1]
                        lnamestaff[del] = lnamestaff[del+1]
                        oneadstaff[del] = oneadstaff[del+1]
                        twoadstaff[del] = twoadstaff[del+1]
                        threeadstaff[del] = threeadstaff[del+1]
                        pcodestaff[del] = pcodestaff[del+1]
                        ninum[del] = ninum[del+1]
                        username[del] = username[del+1]
                        password[del] = password[del+1]
                        loa[del] = loa[del+1]
                End for
        Endif
Endfor
nsi = nsi –1
Do proc ReWriteStaffFile
End proc
```

Change Staff Telephone Number:

```
Proc STelnum
telnum = 0
Do proc ReadBackStaffFile
output “Enter the staff reference: “
input staffref[nsi]
For find = 1 to nsi
        If( find[nsi] = staffref)
                While telnum = 0
                        output “Enter mobile number: “
                        input telnostaff [nsi]
                        telnum = Do Proc TelFormatCk(telnostaff [nsi])
                Endwhile
        End if
End for
Do proc ReWriteStaffFile
End proc
```

Change Staff Home Address:

```
Proc SHomeAdd
presaone = 0, presatwo =0, presathree =0, pcode = 0
Do proc ReadBackStaffFile
output “Enter the staff reference: “
input staffref[nsi]
For find = 1 to nsi
        If( find[nsi] = staffref)
                While presaone = 0
                        output “Enter address line 1: “
```

```
                input oneadstaff [nsi]
                presaone = Do Proc PresenceCk(oneadstaff [nsi])
            Endwhile
            While presatwo = 0
                output “Enter address line 2: “
                input twoadstaff [nsi]
                presatwo = Do Proc PresenceCk(twoadstaff [nsi])
            Endwhile
            While presathree = 0
                output “Enter address line 3: “
                input threeadstaff [nsi]
                presathree = Do Proc PresenceCk(threeadstaff [nsi])
            Endwhile
            While pcode = 0
                output “Enter postcode: “
                input pcodestaff [nsi]
                pcode = Do Proc PostcodeVal(pcodestaff [nsi])
            Endwhile
        End if
End for
Do proc ReWriteStaffFile
End proc
```

<u>View staff by staff reference:</u>

```
Proc ViewStaffRef
output “Enter staff reference: “
input staffref[nsi]
Do proc ReadBackStaffFile
For find = 1 to nsi
        If (find[nsi] = staffref)
                output “First name: “, fnamestaff[nsi]
```

```
        output “Last name: “, lnamestaff[nsi]
        output “Address line 1: “, oneadstaff[nsi]
        output “Address line 2: “, twoadstaff[nsi]
        output “Address line 3: “, threeadstaff[nsi]
        output “Postcode: “, pcodestaff[nsi]
        output “Telephone number: “, telnostaff[nsi]
        output “National Insurance Number: “, ninum[nsi]
        output “Level of access: “, loa[nsi]
End if
Else
        Output “Staff reference does not exist.”
End proc
```

Serial File – Invoices

ReadBackInvoicesFile:

```
Proc ReadBackInvoicesFile
Open InvoicesFile
Read nii
For count = 1 to nii
        read invoiceref(count)
        read custno(count)
        read jobstartdate(count)
        read jobenddate(count)
        read finaldesc(count)
        read matprices(count)
        read mileage(count)
        read totalprice(count)
        read paid(count)
End for
Close InvoicesFile
End proc
```

ReWriteInvoicesFile:

```
Proc ReWriteInvoicesFile
Open InvoicesFile
Write nii
For count = 1 to nii
        write invoiceref(count)
```

```
        write custno(count)
        write jobstartdate(count)
        write jobenddate(count)
        write finaldesc(count)
        write matprices(count)
        write mileage(count)
        write totalprice(count)
        write paid(count)
End for
Close InvoicesFile
End proc
```

Add Invoice:

```
Proc AddInvoice
uniqueref = 0, uniquecust=0, startdate=0, datesyst = 0, enddate=0, presd =0, presnum=0, rangep=0, presmat=0, preslab=0, presmile=0
Do Proc ReadBackInvoicesFile
While uniqueref = 0
        output “Enter new invoice reference: ”
        input invoiceref[nii]
        uniqueref = Do Proc UniqueVal(invoiceref[nii])
Endwhile
While uniquecust= 0
        output “Enter customer reference: “
        input custno [nii]
        uniquecust= Do Proc UniqueVal(custno[nii])
```

```
Endwhile
While startdate= 0 and datesyst = 0
        output “Enter start date: “
        input jobstartdate[nii]
        startdate= Do Proc DateVal(jobstartdate[nii])
        Datesyst = Do Proc SystemsClock(jobstartdate[nii])
Endwhile
While enddate= 0
        output “Enter start date: “
        input jobenddate[nii]
        enddate= Do Proc DateVal(jobenddate[nii])
Endwhile
While presd= 0
        output “Enter job description: “
        input finaldesc[nii]
        presd= Do Proc PresenceCk(finaldesc[nii])
Endwhile
While presnum = 0
        output “Enter number of days worked: “
        input inumofdays[nii]
        presnum = Do Proc PresenceCk(inumofdays[nii])
Endwhile
While presmat= 0
        output “Enter cost of materials: “
        input matprices[nii]
        presmat= Do Proc PresenceCk(matprices[nii])
Endwhile
```

```
While presmile= 0
        output “Enter mileage costs: “
        input mileage[nii]
        presmile= Do Proc PresenceCk(mileage[nii])
Endwhile
While rangep= 0
        output “Is job paid for: ( 0 = Yes,  1 = No) “
        input paid[nii]
        rangep= Do Proc RangeCk(paid[nii])
Endwhile
Do proc icalc (inumofdays, matprices, mileage)
nii = nii + 1
Do Proc ReWriteInvoiceFile
End proc
```

Delete Invoice:

```
Proc DeleteInvoice
Do Proc ReadBackInvoicesFile
Output “Enter the invoice reference: “
input invoiceref[nii]
For find = 1 to nii
        if (find[nii] = invoiceref)
                for del = find to nii
                        invoiceref[del] = invoiceref[del+1]
                        custno[del] = custno[del+1]
                        jobstartdate[del]=jobstartdat[del+1]
                        jobenddate[del] = jobenddate[del+1]
                        finaldesc[del] = finaldesc[del+1]
```

```
                    inumofdays[del] = inumofdays[del+1]
                    matprices[del] = matprices[del+1]
                    mileage[del] = mileage[del +1}
                    totalprice[del] = totalprice[del+1]
                    paid[del] = paid[del+1]
                End for
        Endif
        Else
        Output "Invoice reference does not exist."
Endfor
nii = nii –1
Do proc ReWriteInvoiceFile
End proc
```

Change Invoice Payment Status:

```
Proc ChangePaid
rangep= 0
Do proc ReadBackInvoiceFile
output "Enter the invoice reference: "
input invoiceref[nii]
For find = 1 to nii
        If( find[nii] = invoiceref)
                While rangep= 0
                        output "Enter payment status: "
                        input paid[nii]
                        rangep= Do Proc RangeCk(paid[nii])
                Endwhile
        End if
        Else
```

Output “Invoice reference does not exist.”

End for

Do proc ReWriteInvoicesFile

End proc

Calucating cost:

Proc icalc (inumofdays, matprices, mileage)

Do proc ReadBackInvoicesFile

Labour = inumofdays * 150

cost = labourcost + matprices + mileage

VAT = 0.8 * cost

Totalprice = cost + VAT

output “Break down:”

output “ Material Prices: “, matprices

output “ Labour Prices: “, labour

output “ Mileage: “, mileage

output “ VAT: “, VAT

output “ Total price:”, totalprice

return totalprice

End proc

View invoice by invoice reference:

```
Prov ViewIRef
output “Enter invoice reference: “
input inviceref[nii]
Do proc ReadBackInvoicesFile
For find = 1 to nii
        If (find[nii] = invoiceref)
                output “ Customer reference: “, custno[nii]
                Do proc LocateCustomer(custno)
                output “Starting date of the job : “, jobstartdate[nii]
                output “ End date of the job: “, jobenddate[nii]
                output “ Job description: “, finaldesc[nii]
                Do proc ICalc (inumofdays[nii], matprices[nii], mileage[nii])
        End if
        Else
                Output “Invoice reference does not exist.”
End for
End proc
```

Locate Customer:

```
Proc LocateCustomer(custno)
Locate(custref)
If (flag!=0)
                output “Customer name: “, a_cust.fnamecust, a_cust.lnamecust
                output “Telephone number: “, a_cust.telnocust
End if
Return custno
End proc
```

Search for unpaid invoices:

```
Proc SearchUnpaid
ReadBackInvoicesFile
For find = 1 to nii
        (if paid == 0)
                output “Customer Reference”, custno
        End if
        Else
                output “All invoices are paid for.”
End for
End proc
```

Serial File – Quotes

ReadBackStaffFile:

```
Proc ReadBackQuotesFile
Open QuotesFile
Read nqi
For count = 1 to nqi
        read quoteref(count)
        read custno(count)
        read mainjobdesc(count)
        read eststock(count)
        read qnumofdays(count)
        read matprices(count)
        read totalprice(count)
End for
Close QuotesFile
End proc
```

ReWriteQuotesFile:

```
Proc ReWriteQuotesFile
Open QuotesFile
Write nqi
For count = 1 to nqi
        write quoteref(count)
        write custno(count)
        write mainjobdesc(count)
```

```
        write eststock(count)
        write qnumofdays(count)
        write matprices(count)
        write totalprice(count)

End for
Close QuotesFile
End proc
```

Add Quote:

```
Proc AddQuote
unique = 0, uniquecust, presm, presnum, mile
Do proc ReadBackQuoteFile
While unique = 0
        output “Enter new quote reference: ”
        input quoteref [nqi]
        unique = Do Proc UniqueVal(quoteref[nqi])
Endwhile
While uniquecust = 0
        output “Enter customer reference: “
        input custno[nqi]
        uniquecust = Do proc FindCustomer(custno[nqi])
Endwhile
While presm = 0
        output “Enter description of the job”
        input mainjobdesc[nqi]
        presm = Do Proc PresenceCk(mainjobdesc[nqi])
Endwhile
```

```
While presnum = 0
        output “Enter the number of days you will spend on the job”
        input qnumofdays[nqi]
        presnum = Do Proc PresenceCk(qnumofdays[nqi])
Endwhile
While mile = 0
        output “Enter travel costs: “
        input mileage[nqi]
        mile = Do Proc RangeCk(mileage[nqi])
Endwhile
Stockcost = Do proc StockCalc
Labour = qnumofdays * 150
Input labour[nqi]
Pricenovat = total + labour+ mileage
VAT = 0.8*pricenovat
Input vat[nqi]
Totalcost = total + labour+ mileage + VAT
input totalcost[nqi]
output “Break down:”
output “                Material Prices: “, total
output “                Labour Prices: “, labour
output “                Mileage: “, mileage
output “                VAT: “, VAT
output “               Total price:”, totalcost

ReWriteQuotesFile
End proc

Proc StockCalc
```

```
Do Proc ReadBackLinksFile
output “How many items of stock are required”
input numofitems
For add = 1 to numofitems
        output “ Enter stock ID: “
        input stockno
        Cost = Do proc FindStock (stockno)
        Total = cost + total
End for
Return total
End proc


Proc FindStock(stockref)
Do proc ReadBackStockFile
For find = 1 to nsti
        If (find[nsti] = stockref)
        cost = stockprice[nsti]
        End if
End for
Return cost
End proc
```

Delete a quote:

```
Proc DeleteQuote
Do proc ReadBackQuotesFile
output “Enter the quotereference: “
```

```
input quoteref[nqi]
for find = 1 to nqi
        if (find[nqi] = quoteref)
                for del = find to nqi
                quoteref[del] = quoteref[del+1]
                custno[del] = custno[del+1]
                mainjobdesc[del] = mainjobdesc[del+1]
                qnumofdays[del] = qnumofdays[del+1]
                mileage[del] = mileage[del+1]
                totalcost[del] = totalcost [del+1]
                End  for
        End if
End for
nqi = nqi –1
Do proc ReWriteQuotesFile
End proc
```

Change total price of quote:

```
Proc ChangePrice
Do Proc ReadBackQuotesFile
output “Enter quote reference: “
input quoteref[nqi]
For find = 1 to nqi
        If(find[nqi] = quoteref)
                while pres = 0 and range = 0
                        output “Enter updated price: “
                        input totalcost[nqi]
                        pres = Do Proc PresenceCk(totalcost(nqi))
```

```
                range = Do Proc RangeCk(totalcost(nqi))
            end while
        End if
        Else
        output “Quote reference does not exist on the system.”
End for
Do proc ReWriteQuotesFile
End proc
```

Change number of days on the quote:

```
Proc ChangeDays
Do Proc ReadBackQuotesFile
output “Enter quote reference: “
input quoteref[nqi]
For find = 1 to nqi
        If(find[nqi] = quoteref)
            while range = 0
                output “Enter number of days expected to work: “
                input qnumofdays[nqi]
                range = Do Proc RangeCk(qnumofdays(nqi))
            end while
        End if
        Else
            output “Quote reference does not exist on the system.”
End for
Do proc ReWriteQuotesFile
End proc
```

View quote by quote reference:

```
Prov ViewQuoteRef
output “Enter quote reference: “
Input quoteref[nqi]
Do proc ReadBackQuotesFile
For find = 1 to nqi
        If (find[nqi] = quoteref)
                output “ Customer reference: “, custno[nqi]
                Do proc LocateCustomer(custno)
                output “Job description: “ , mainjobdesc[nqi]
                output “Number of days to be worked: “ , qnumofdays[nqi]
                output “ Price Break down:”
                output “                Material Prices: “, stockprice[nqi]
                output “                Labour Prices: “, labour[nqi]
                output “                Mileage: “, mileage[nqi]
                output “                VAT: “, vat[nqi]
                output “               Total price:”, totalcost[nqi]
        End if
        Else
                Output “Quote reference does not exist.”
End for
End proc
```

View Quote by customer reference:

```
Prov ViewQuotebyCustRef
output “Enter customer reference: “
input custno[nqi]
Do proc ReadBackQuotesFile
For find = 1 to nqi
```

```
        If (find[nqi] = custno)
                Do proc LocateCustomer(custno)
                output " Quote reference: ", quoteref[nqi]
                output "Job description: " , mainjobdesc[nqi]
                output "Number of days to be worked: " , qnumofdays[nqi]
                output " Price Break down:"
                output "                Material Prices: ", stockprice[nqi]
                output "                Labour Prices: ", labour[nqi]
                output "                Mileage: ", mileage[nqi]
                output "                VAT: ", vat[nqi]
                output "               Total price:", totalcost[nqi]
        End if
        Else
                Output "Quote reference does not exist."
End for
End proc
```

Locate Customer:

```
Proc LocateCustomer(custno)
Locate(custref)
If (flag!=0)
                output "Customer name: ", a_cust.fnamecust, a_cust.lnamecust
                output "Telephone number: ", a_cust.telnocust
End if
Return custno
End proc
```

Sequential File – Schedule

ReadBackBookingSchedule:

```
Proc ReadBackBookingSchedule
Open BookingScheduleFile
for staffno = 1 to 2
        for date = 1 to 365
                for hour = 1 to 12 (7  to 19)
                        read back bookingsch[staffno][date][hour]
                endfor
        endfor
endfor
Close BookingScheduleFile
End proc
```

ReWriteBookingSchedule:

```
Proc ReWriteBookingSchedule
Open BookingScheduleFile
for staffno = 1 to 2
        for date = 1 to 365
                for hour = 1 to 12
                        rewrite bookingsch[staffno][date][hour]
                endfor
        endfor
endfor
Close BookingScheduleFile
End proc
```

Add Booking:

```
Proc AddBooking
Open BookingScheduleFile
Do Proc ReadBackBookingSchedule
Output “Enter staff reference: “
Input staffno
Output “Enter date of the booking: ”
Input date
Output “Enter hour: “
Input hour
Output “Enter quote reference: “
Input quoteno
ReWrite Bookingsch([staffno][date][hour], quoteno)
Do Proc ReWriteBooking Schedule
Close BookingScheduleFile
End proc
```

Delete booking:

```
Proc DeleteBooking
Do proc ReadBackBookingSchedule
Output “Enter staff reference for the booking”
Input staffno
Output “Enter the number of days on the booking: “
Input numday
For del = 1 to numday
        Output “Enter the date of the booking: “
        Input date
        Output “Enter the number of hours worked that day: “
        Input numhour
```

```
        For time = 1 to numhour
                Output “Enter first hour on the booking”
                input hour
                ReWriteBookingSch([staffno][date][hour], “ “)
        End for
End for
```

View schedule:

```
Proc ViewSchedule
Do proc ReadBackBookingSchedule
Output “Enter the staff reference to view their schedule: “
Input staffno
For date = 1 to 365
        Output days[date]
        For hour = 1 to 12
                output bookingsch[staffno][date][hour]
        End for
End for
End proc
```

Change Schedule dates:

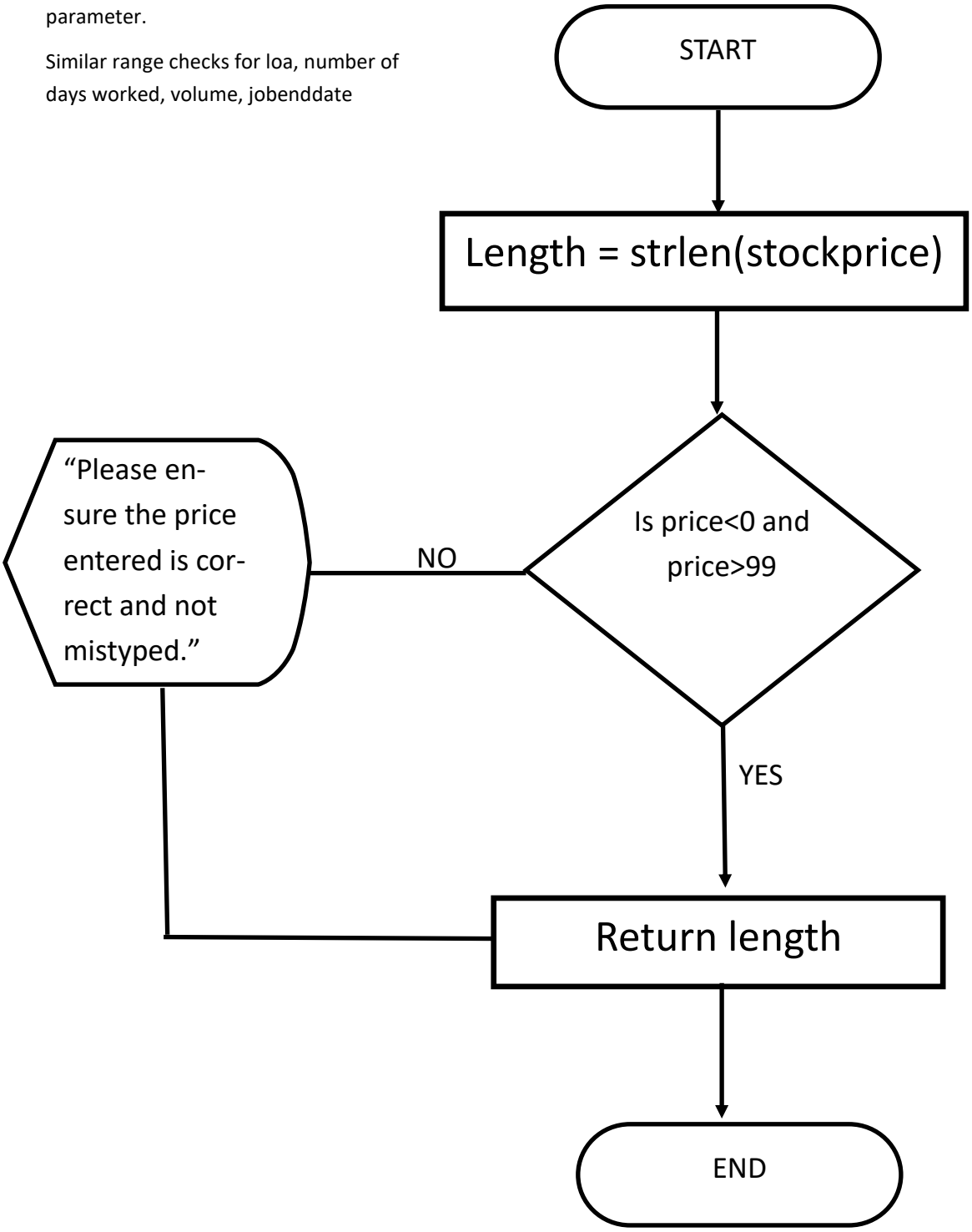
```
Proc ChangeDates

Do proc ReadBackBookingSchedule
Output “Enter the staff reference of the booking you wish to change”
Input staffno
Output “Enter the number of days on the booking: “
Input numday
```

```
For del = 1 to numday
        Output “Enter the date of the booking: “
        Input date
        Output “Enter the number of hours worked that day: “
        Input numhour
        For time = 1 to numhour
                Output “Enter first hour on the booking”
                input hour
                Output “Enter quote reference: “
                Input quoteno
                ReWriteBookingSch([staffno][date][hour], quoteno)
        End for
End for
End proc
```

Proc RangeCk

Variable 'stockprice' is passed through as a parameter.

Similar range checks for loa, number of days worked, volume, jobenddate



Random Access – Customers

View customer by customer reference:

```
Proc ViewbyCustNum
ouput “Enter the customer number: “
input custref
Open CustomerFile
Locate(custref)
Output “Customer name: ” , a_cust.fnamecust a_cust.lnamecust
Output “Address line 1: “ , a_cust.oneadcust
Output “Address line 2: “ , a_cust.twoadcust
Output “Address line 3: “ , a_cust.threeadcust
Output “Postcode: “ , a_cust.pcodecust
Output “Telephone number: “ , a_cust.telnocust
End proc
```

View customer by customer name:

```
Proc ViewbyCustName
output “Enter customer last name: “
input lnamecust
Open CustomerFile
Locate(lnamecust)
For find = 1 to nci
        if (a_cust.lnamecust = lnamecust)
                output “First name: ” , a_cust.fnamecust
                output “Last name: “ , a_cust.lnamecust
                output “Address line 1: “ , a_cust.oneadcust
                output “Address line 2: “ , a_cust.twoadcust
                output “Address line 3: “ , a_cust.threeadcust
```

```
            output “Postcode: “ , a_cust.pcodecust
            output “Telephone number: “ , a_cust.telnocust
        End if
End for
End proc
```

Add Customer:

```
Proc AddCust
presf = 0, presl =0, presaone = 0, presatwo = 0, presathree = 0, pcode =0, telnum=0
output “Enter a new customer number: “
input custref
Open CustomerFile
If (flag=0)
        While presf = 0
            output “Enter first name: “
            input a_cust.fnamecust
            presf = Do Proc PresenceCk(fnamecust)
        Endwhile
        While presl = 0
            output “Enter last name: “
            input a_cust.lnamecust
            presl = Do Proc PresenceCk(lnamecust)
        Endwhile
        While presaone = 0
            output “Enter address line 1: “
            input a_cust.oneadcust
            presa1 = Do Proc PresenceCk(oneadcust)
        Endwhile
        While presatwo = 0
            output “Enter address line 2: “
            input a_cust.twoadcust
```

```
            presatwo = Do Proc PresenceCk(twoadcust)
        Endwhile
        While presathree = 0
            output “Enter address line 3: “
            input a_cust.threeadcust
            presa3 = Do Proc PresenceCk(threeadcust)
        Endwhile
        While pcode = 0
            output “Enter postcode: “
            input a_cust.pcodecust
            pcode = Do Proc PostcodeVal(pcodecust)
        Endwhile
        While telnum = 0
            output “Enter telephone number: “
            input a_cust.telnocust
            telnum = Do Proc TelFormatCk(telnocust)
        Endwhile
        flag = 1
End if
Else
        Output “Stock reference is already in use”
End proc
```

Delete Customer:

```
Proc DelCustomer
output “Enter customer reference: “
input custref
Open CustomerFile
Locate (custref)
If(flag=1)
        a_cust.fnamecust = a_cust.fnamecust + 1
```

```
        a_cust.lnamecust = a_cust.lnamecust + 1
        a_cust.1adcust = a_cust.oneadcust + 1
        a_cust.2adcust = a_cust.twoadcust + 1
        a_cust.3adcust = a_cust.threeadcust + 1
        a_cust.pcodecust = a_cust.pcodecust + 1
        a_cust.telnocust = a_cust.telnocust + 1
End if
Else
        Output “Customer reference does not exist within the system.”
flag = flag –1
End proc
```

Change customer address:

```
Proc ChangeCustomerH
presa=0, presone=0, presatwo=0, presathree=0, pcode=0
output “Enter customer reference: “
input custref
Open CustomerFile
Locate (custref)
If(flag=1)
        While presaone = 0
                output “Enter address line 1: “
                input a_cust.oneadcust
                presaone = Do Proc PresenceCk(oneadcust)
        Endwhile
        While presatwo = 0
                output “Enter address line 2: “
                input a_cust.twoadcust
                presatwo = Do Proc PresenceCk(twoadcust)
```

```
        Endwhile
        While presathree = 0
                output “Enter address line 3: “
                input a_cust.threeadcust
                presathree = Do Proc PresenceCk(threeadcust)
        Endwhile
        While pcode = 0
                output “Enter postcode: “
                input a_cust.pcodecust
                pcode = Do Proc PostcodeVal(pcodecust)
        Endwhile
End if
Else
        Ouput “Customer reference does not exist on the system.”
End proc
```

Change customer telephone number:

```
Proc ChangeCustTelno
telnum=0
output “Enter customer reference: “
input custref
Open CustomerFile
Locate (custref)
If(flag=1)
        While telnum = 0
                output “Enter telephone number: “
                input a_cust.telnocust
                telnum = Do Proc TelFormatCk(telnocust)
        Endwhile
Endif
```

Else

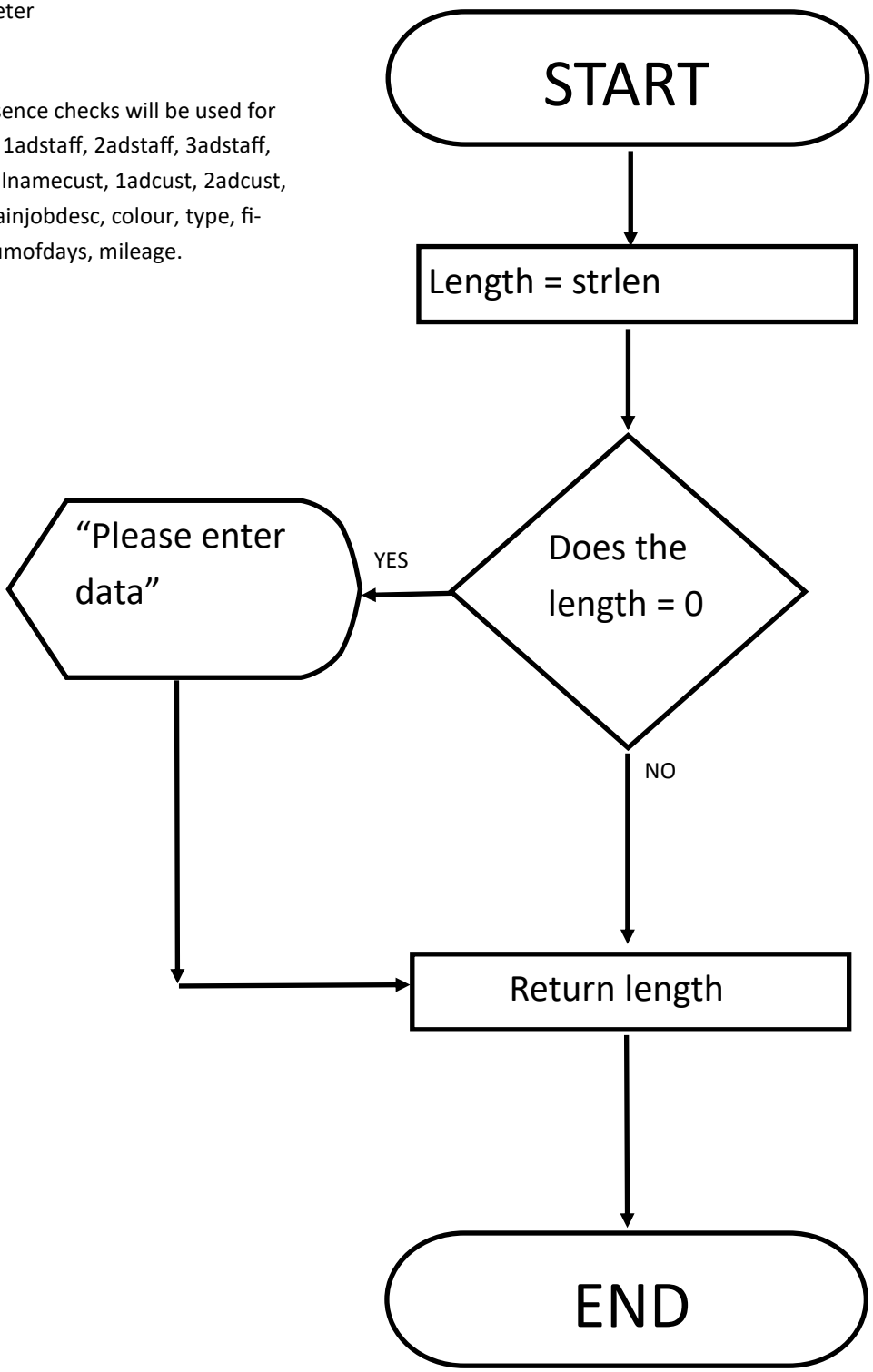
    Ouput “Customer reference does not exist on the system.”

End proc

Proc PresenceCk

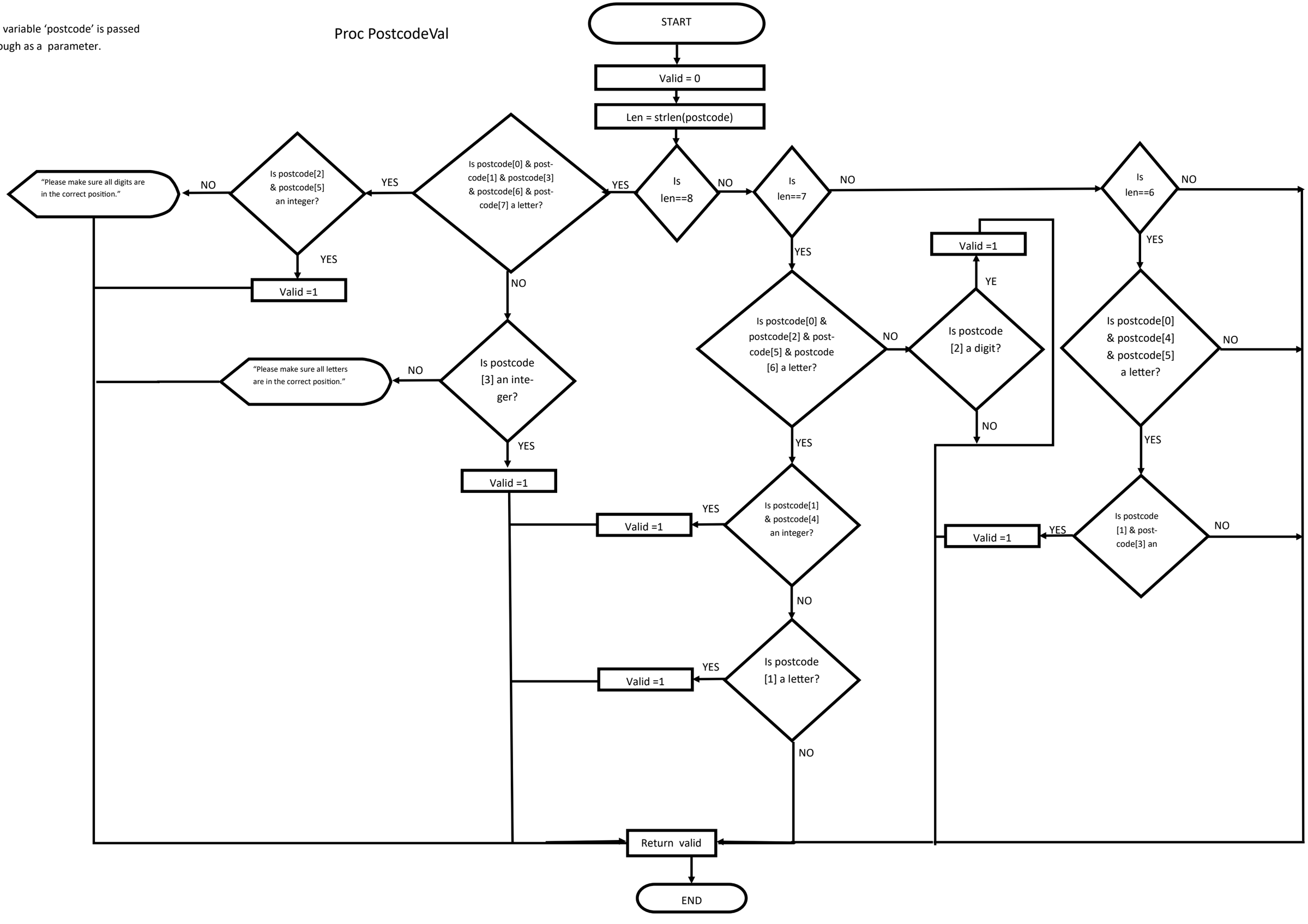
The variable ‘fnamestaff’ is passed through as a parameter

Similar presence checks will be used for lnamestaff, 1adstaff, 2adstaff, 3adstaff, fnamecust, lnamecust, 1adcust, 2adcust, 3adcust, mainjobdesc, colour, type, finaldesc, inumofdays, mileage.



The variable ‘postcode’ is passed through as a parameter.

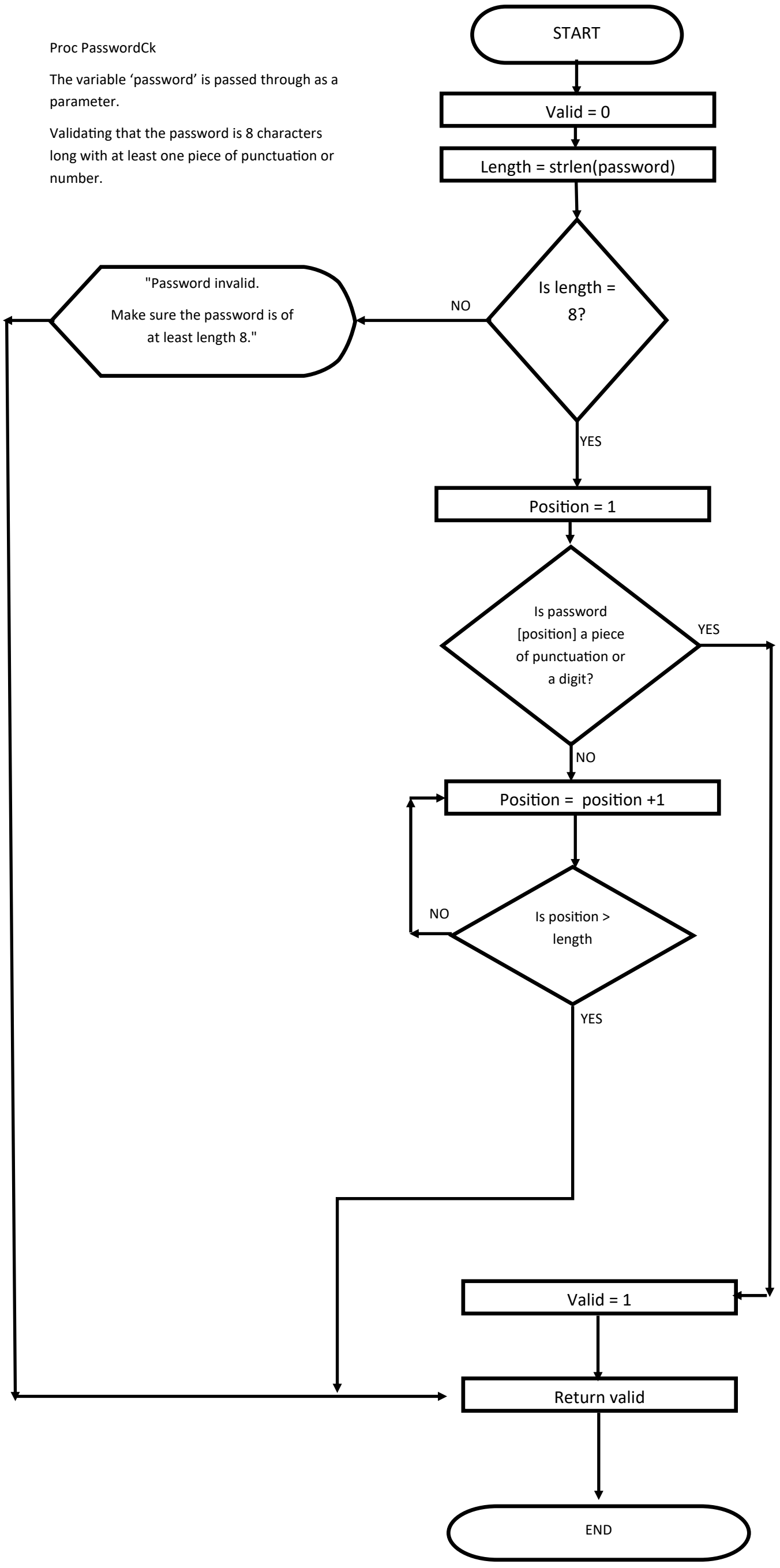
Proc PostcodeVal



Proc PasswordCk

The variable ‘password’ is passed through as a parameter.

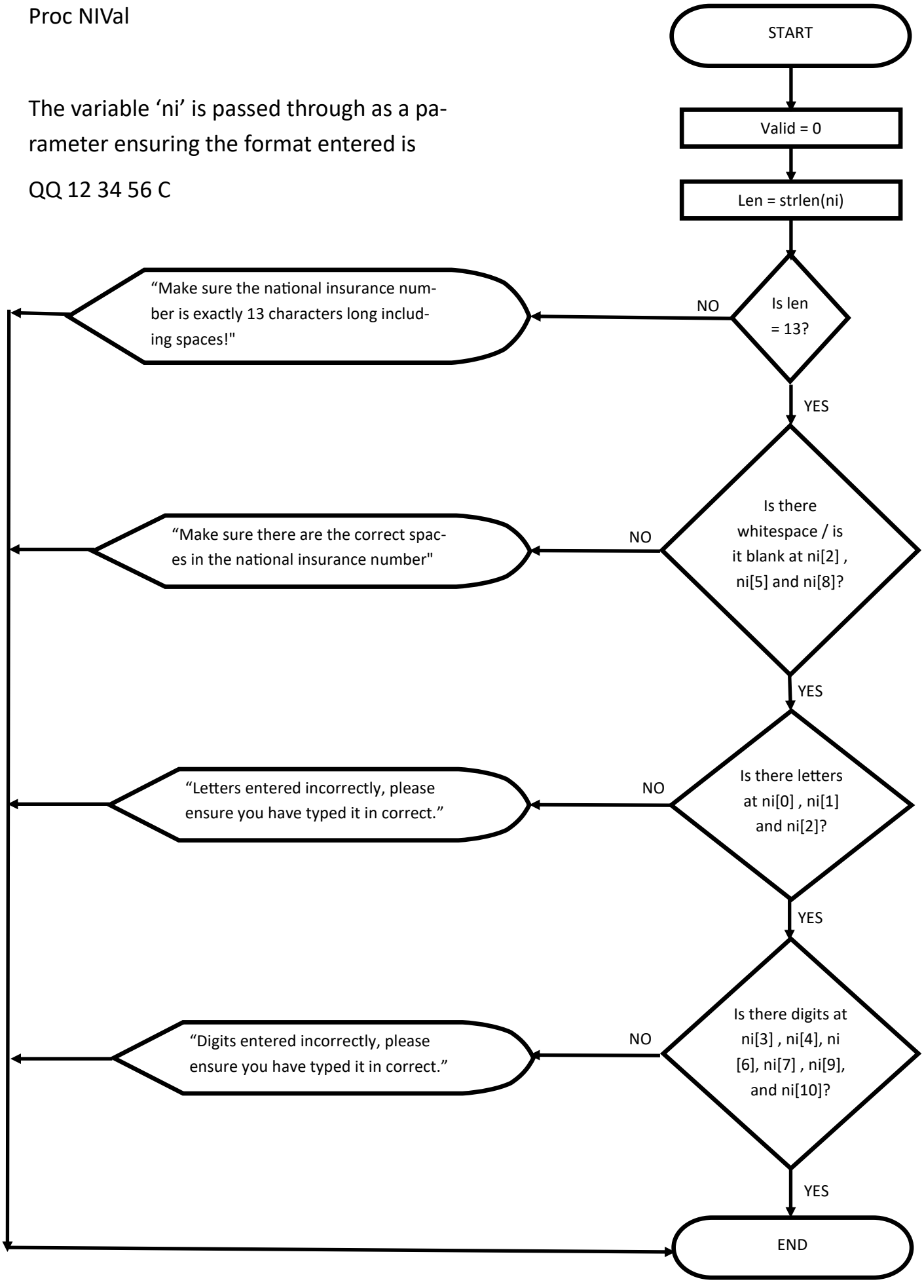
Validating that the password is 8 characters long with at least one piece of punctuation or number.



Proc NIVal

The variable ‘ni’ is passed through as a parameter ensuring the format entered is

QQ 12 34 56 C



Logging On

```
Proc Login

Do Proc ReadBackStaffFile
if(nsi !=0 && username[0] = “Admin” && pass[0] = “Pass”)
        firsttime = 0
        do proc Username
End if
Else
output “Enter new username: “
input username[nsi]
output “Enter new password: “
input password[nsi]
output “Enter level of access: “
input loa[nsi]
nsi = nsi + 1
Do Proc ReWriteStaffFile
End proc

Proc Username

Do proc ReadBackStaffFile
user = 0
While user = 0
output “Enter username: “
input username[nsi]
user = Do Proc UsernameValid(username[nsi])
End while
```

```
End proc

Proc UsernameValid (username)
Do proc ReadBackStaffFile
For find = 1 to nsi
If(find[nsi] = username)
        Do proc Password
    End if
    Else
    output “Username not correct”
End for
End proc

Proc Password
pass = 0
Do proc ReadBackStaffFile
While pass= 0
output “Enter password: “
input password[nsi]
user = Do Proc PassValid(password[nsi])
End while
End proc

Proc PassValid (password)
Do proc ReadBackStaffFile
For find = 1 to nsi
If(find[nsi] = password)
        Do proc MainMenuType
    End if
```

```
        Else
        output “Password not correct”
End for
End proc

Proc MainMenuType

Do Proc ReadBackStaffFile
For find = 1 to nsi
If (find[nsi]=username)
if (loa[nsi] = 1)
Do Proc MainMenuWhileOne
end if
                elif (loa[nsi] = 2)
Do proc MainMenuWhileTwo
end if
                Else
                        If (loa[nsi]=3)
                                Do Proc MainMenuWhileThree
                        End if

End if
End for
End proc

Proc MainMenuWhileOne
mainstatus = 0
while(mainstatus =0)
mainstatus = Do Proc MainMenuOne
```

End while

End proc

Proc MainMenuOne

only has the option to view information, cannot make any editorial decisions apart from in the customer, quote and invoice section. However does not have the option to change the payment status of an invoice this must be done by somebody higher up.

int choice

output “Main Menu”

output “1. Customers”

output “2. Quotes”

output “3. Invoices”

output “4. Stock”

output “5. Schedule”

output “Enter choice”

input choice

switch(choice)

Case 1:

    Do proc Customers

    break

Case 2:

    Do proc Quotes

    break

Case 3:

    Do proc Invoices

    break

Case 4:

    Do proc Stock

    break

```
        Case 5:
                Do proc Schedule
                break
        Case 6:
                Break
        End case
End Proc

Proc MainMenuWhileTwo
mainstatus = 0
while(mainstatus =0)
mainstatus = Do Proc MainMenuTwo
End while
End proc

Proc MainMenuTwo
# Has the ability to change all areas of the system shown below apart from changes to the schedule
int choice
output “Main Menu”
output “1. Customers”
output “2. Quotes”
output “3. Invoices”
output “4. Stock”
output “5. Schedule”
output “Enter choice”
input choice
switch(choice)
Case 1:
                Do proc Customers
```

```
            break
        Case 2:
            Do proc Quotes
            break
        Case 3:
            Do proc Invoices
            break
        Case 4:
            Do proc Stock
            break
        Case 5:
            Do proc Schedule
            break
        Case 6:
            Break
        End case
End Proc

Proc MainMenuWhileThree
mainstatus = 0
while(mainstatus =0)
mainstatus = Do Proc MainMenuThree
End while
End proc

Proc MainMenuThree

int choice
output “Main Menu”
```

```
output “1. Staff”
output “2. Customers”
output “3. Quotes”
output “4. Invoices”
output “5. Stock”
output “6. Schedule”
output “Enter choice”
input choice
switch(choice)
Case 1:
Do proc Staff
break
Case 2:
        Do proc Customers
        break
    Case 3:
        Do proc Quotes
        break
    Case 4:
        Do proc Invoices
        break
    Case 5:
        Do proc Stock
        break
    Case 6:
        Do proc Schedule
        break
    Case 7:
        Break
```

```
        End case
End Proc
```

Proc TelFormatCk

Validation check to ensure the format of the mobile number entered is 0N=7NNNNNNNNN

Variable ‘phonenum’ is passed through as a parameter