**Q1.**

**TweetApp.cpp**

#include <string> //include string datatype so i can use it throughout the file

#include "Tweet.h";//include header file of tweet

#include <iostream>//allow input and output of program

using namespace std; //allows the use of classes objects and functions under a name

int main()//this is where the program executes

{

Tweet \* t1 = new Tweet("shawn", "hello", "12 Dec 2012", false);// create Tweet object name t1 and initiliase the vars of the object.

Tweet \* t2 = new Tweet("shawn", "ya wats up", "12 Dec 2012", false);// create Tweet object name t2 and initiliase the vars of the object.

cout << t1->toString() << endl; //output t1 object point to toString function which returns a string object

cout << t2->toString() << endl; //output t2 object point to toString function which returns a string object

}

**Tweet.cpp**

#include <string> //include string so i can use string data type throughout

#include "Tweet.h"; //include Tweet header file

#include <iostream> //allow input and output of program

#include <sstream> //allows the use of stringstream data type

using namespace std; //allows the use of classes objects and functions under a name

Tweet::Tweet(string username, string tweet\_text, string tweet\_date, bool media) //tweet constructor

{

this->username = username; //initalises all values. the this keyword is necessary as both vars have same name.

this->tweet\_text = tweet\_text;

this->tweet\_date = tweet\_date;

this->media = media;

}

string Tweet::toString()//to string function declaration

{

stringstream info;

info << "@" << username << " " << tweet\_text << " " << tweet\_date << endl;//passes the whole string into info var.

return info.str();//return as a string variable.

}

**Tweet.h**

#include <string> //include string so i can use string data type throughout

using namespace std;//allows the use of classes objects and functions under a name

class Tweet //Creation of Class Tweet

{

private://the variables in this class can only be accessed by this class.

string username;

string tweet\_text;

string tweet\_date;

bool media;

public://any class can call the functions in this class.

Tweet(string username, string tweet\_text, string tweet\_date, bool media);

string toString();

};

**Q2.**

**RobotApp.cpp**

#include "Robot.h" //include Robot Headerfile

#include <string> //include string so i can use the string data type

#include <cstdlib> //for srand to work

#include <ctime> //to manipulate time and data

#include <iostream> //allows input and output of the program

using namespace std; //allows the use of classes objects and functions under a name

int main()

{

srand(time(0)); //initialise random number generator

Robot \* r1 = new Robot("dummy",0,0,"east"); //create Robot object with preset values

int randomx = rand() % 100 + 1; //range from 1 to 100 random number

int randomy = rand() % 100 + 1; //range from 1 to 100 random number

Robot \* r2 = new Robot("robot",randomx,randomy,"east"); //create Robot object with preset values

cout << "Current X Coordinates: " << r2->get\_x\_loc() << endl; //using 2nd robot object point to function to get x location

cout << "Current Y Coordinates: " << r2->get\_y\_loc() << endl; //using 2nd robot object point to function to get y location

cout << "Direction: " << r2->get\_direction() << endl; //using 2nd robot object to point to get direction

r2->leftTurn();//r2 point to left turn to turn left

cout << "Direction: " << r2->get\_direction() << endl;//r2 point to get direction

r2->rightTurn();//r2 point to right turn to turn right

cout << "Direction: " << r2->get\_direction() << endl;//r2 point to get direction

int random\_dist = rand() % 10 + 1; //range from 1 to 10

r2->move(random\_dist);//r2 moves at a random distance which is generated frm above

cout << "Current X Coordinates: " << r2->get\_x\_loc() << endl; //output r2 x location which is a int

cout << "Current Y Coordinates: " << r2->get\_y\_loc() << endl;//output r2 y location which is a int

cout << "Direction: " << r2->get\_direction() << endl;//output the current direction of r2

}

**Robot.h**

#include <string>//include string so i can use the string data type

using namespace std; //allows the use of classes objects and functions under a name

class Robot // Robot class created

{

private://the variables in this class can only be accessed by this class.

string robot\_name;

int robot\_loc\_x;

int robot\_loc\_y;

string robot\_direction;

public://any class can call the functions in this class.

Robot(string robot\_name,int robot\_loc\_x,int robot\_loc\_y,string robot\_direction); //constructor

void move(int distance);//void because no return of value

void leftTurn();

void rightTurn();

int get\_x\_loc();

int get\_y\_loc();

string get\_direction();

};

**Robot.cpp**

#include "Robot.h" //robot.h

#include <string> // allows string data type to be used

using namespace std; //allows the use of classes objects and functions under a name

Robot::Robot(string robot\_name,int robot\_loc\_x,int robot\_loc\_y,string robot\_direction) //robot constructor

{

this->robot\_name = robot\_name;//initialise the variables of the object that will be created

this->robot\_loc\_x = robot\_loc\_x;

this->robot\_loc\_y = robot\_loc\_y;

this->robot\_direction = robot\_direction;

}

void Robot::move(int distance)//move function with distance

{

if (robot\_direction == "north") //check if the variable is equals to north and so on and so forth

{

robot\_loc\_y += distance;// add distance to the y location of robot var

}

else if (robot\_direction == "south")

{

robot\_loc\_y -= distance; // deduct distance to the y location of robot var

}

else if (robot\_direction == "west")

{

robot\_loc\_x -= distance; // deduct distance to the x location of robot var

}

else if (robot\_direction == "east")

{

robot\_loc\_x += distance; // add distance to the x location of robot var

}

}

void Robot::leftTurn()

{

if (robot\_direction == "north") //self explanatory

{

robot\_direction = "west";

}

else if (robot\_direction == "south")

{

robot\_direction = "east";

}

else if (robot\_direction == "west")

{

robot\_direction = "south";

}

else if (robot\_direction == "east")

{

robot\_direction = "north";

}

}

void Robot::rightTurn()

{

if (robot\_direction == "north")

{

robot\_direction = "east";

}

else if (robot\_direction == "south")

{

robot\_direction = "west";

}

else if (robot\_direction == "west")

{

robot\_direction = "north";

}

else if (robot\_direction == "east")

{

robot\_direction = "south";

}

}

int Robot::get\_x\_loc()

{

return robot\_loc\_x;

}

int Robot::get\_y\_loc()

{

return robot\_loc\_y;

}

string Robot::get\_direction()

{

return robot\_direction;

}

**Q3.**

**singleAddress.h**

#include <string> //allows the use of string datatype

using namespace std; //allows the use of classes objects and functions under a name

class SingleAddress

{

private: //the variables in this class can only be accessed by this class.

string lastName;

string firstName;

string streetAddress;

string city;

string country;

int postalCode;

int homePhone;

int mobilePhone;

string email;

public://any class can call the functions in this class.

SingleAddress(string lastName,string firstName,string streetAddress,string city,string country,int postalCode,int homePhone,int mobilePhone,string email); //constructor

string toString();

string getlastName();

void setLastName(string lastName);

string getStreetAddress();

};

**singleAddress.cpp**

#include <string>//allows the use of string datatype

#include "SingleAddress.h"//include singleAddress header file

#include <sstream> //allows use of ifstream and ofstream

using namespace std; //allows the use of classes objects and functions under a name

SingleAddress::SingleAddress(string lastName,string firstName,string streetAddress,string city,string country,int postalCode,int homePhone,int mobilePhone,string email)

{

this->lastName = lastName; //initialise the variables of the object that will be created

this->firstName = firstName;

this->streetAddress = streetAddress;

this->city = city;

this->country = country;

this->postalCode = postalCode;

this->homePhone = homePhone;

this->mobilePhone = mobilePhone;

this->email = email;

}

string SingleAddress::toString()

{

stringstream info;

info << "First Name : " << firstName << endl; //stores everything in to stringstream variable..

info << "Last Name : " << lastName << endl;

info << "Street Address : " << streetAddress << endl;

info << "City : " << city << endl;

info << "Postal Code : " << postalCode << endl;

info << "Mobile Phone Number : " << mobilePhone << endl;

info << "Home Phone Number : " << homePhone << endl;

info << "Email : " << email << endl;

return info.str(); //return stringstream to a string datatype

}

string SingleAddress::getlastName() //getlastname method

{

return lastName;

}

void SingleAddress::setLastName(string lastName) //lastName parameter which will be passed here

{

this->lastName = lastName; //set lastName of that object to the lastName value that is passed here

}

string SingleAddress::getStreetAddress()

{

return streetAddress;

}

**AddressBookApp**

#include <iostream> //allows input and output

#include "AddressBook.h" //include AddressBook headerfile

#include <vector> //allows use of vector

using namespace std; //allows the use of classes objects and functions under a name

//global vars

vector<SingleAddress\*> address\_list;//address\_list is the vector name of singleAddress datatype

AddressBook \* addressbookobj = new AddressBook(address\_list);

int main()

{

//for initialisation give the variables values so that they are initialised.

int choice = 0;

string lastName = " ";

string firstName= " ";

string streetAddress= " ";

string city= " ";

string country= " ";

int postalCode= 0;

int homePhone= 0;

int mobilePhone= 0;

string email= " ";

bool result;

string lastName2 = " ";

string info;

cout << "Main Menu" << endl; //output of the menu

cout << "---------"<< endl;

cout << "1. Add an address"<< endl;

cout << "2. Remove an address"<< endl;

cout << "3. Print all addresses"<< endl;

cout << "4. Quit"<< endl;

cout << endl;

cout << "Enter your choice:"<< endl;

cin >> choice ; //recieves input from the user as to which choice he will choose

if (choice == 1)

{

cout << "Add address selected" << endl;

//codes to add address

cout << "Last Name: "; //cout is for outputting something in this case a string

cin >> lastName;//cin is for recieving input

cout << "First Name: ";

cin >> firstName;

cout << "Street Address: ";

cin >> streetAddress;

cout << "City: ";

cin >> city;

cout << "Country: ";

cin >> country;

cout << "Postal Code: ";

cin >> postalCode;

cout << "Home Phone: ";

cin >> homePhone;

cout << "Mobile Phone: ";

cin >> mobilePhone;

cout << "Email: ";

cin >> email;

cout << endl;

SingleAddress \* singleaddressobj = new SingleAddress(lastName,firstName,streetAddress,city,country,postalCode,homePhone,mobilePhone,email); //create a singleaddress object

addressbookobj->addAddress(singleaddressobj);//points the addressbook obj to addAddress so as to add singleAddress Object into addressBook Object.

main(); // go back to main menu

}

else if (choice == 2)

{

cout << "Remove address selected" << endl;

//codes to remove address

lastName2 = " "; //initialise

cout << "Last Name: ";

cin >> lastName2;

result = addressbookobj->removeAddress(lastName2);//result return type is a bool so its used to store the return of removeAddress function which is also a bool

//addressBook Object points to removeAddress function and passes the lastName2 parameter which is entered by the user.

if (result == true)

{

cout << "Address successfully removed!" << endl;

main(); // go back to main menu

}

else if (result == false)

{

cout << "Address was not found!" << endl;

main(); // go back to main menu

}

}

else if (choice == 3)

{

cout<<"list of address:"<<endl;

cout<<addressbookobj->returnListOfAddresses()<<endl; //calls the return address function returns string

main(); // go back to main menu

}

else if (choice ==4)

{

cout << "Are you sure you want to Quit?" << endl;

cout << "Y for Yes and N for No" << endl;

char prompt; // use char cause its only one character needed

cin >> prompt;

if (prompt == 'y' || prompt == 'Y') //upper or lower case accepted

{

exit;//exit program

}

else if (prompt == 'n'|| prompt == 'N')

{

main();

}

else

{

cout << "Invalid choice. Please try again." << endl;

main();

}

}

else //if user entered numbers other than 1 to 4

{

cout << "Invalid choice. Please try again." << endl;

main();

}

}

**AddressBook.h**

#include <string> //allows the use of string datatype

#include <vector> //allows the use of a vector

#include "singleAddress.h" //includes singleAddress headerfile

using namespace std; //allows the use of classes objects and functions under a name

class AddressBook //creation of AddressBook class

{

private://the variables in this class can only be accessed by this class.

vector<SingleAddress\*> address\_list; //address\_list is the name of the vector of singleAddress as datatype

public://any class can call the functions in this class.

AddressBook(vector<SingleAddress\*> address\_list);

void addAddress(SingleAddress \* parent); //you dont need to return anything so its void

bool removeAddress(string lastName); //returns bool as you need to know if the removing of address is successful

string returnListOfAddresses();//returns string

string searchAddress(string lastName);//lastName as parameter

};

**AddressBook.cpp**

#include <string>//allows the use of string datatype

#include "AddressBook.h"//includes AddressBook headerfile

#include <vector>//allows the use of a vector

#include <sstream>//allows the use of stringstream datatype

using namespace std; //allows the use of classes objects and functions under a name

stringstream info;

AddressBook::AddressBook(vector<SingleAddress\*> address\_list)//constructor

{

this->address\_list = address\_list; //set the parameter address\_list to the object address\_list.this keyword is used because same name.

}

void AddressBook::addAddress(SingleAddress \* parent) //parent is the variable name of SingleAddress datatype using object composition

{

address\_list.push\_back(parent);//push the parameter i got to the address\_list vector

}

bool AddressBook::removeAddress(string lastName)//lastName as the input of the parameter

{

for (int i =0; i<address\_list.size(); i++)//traverse through address\_list vector

{

if (address\_list[i]->getlastName() == lastName)//check every element and gets lastName and checks if the user input lastName is the same as the current element it is looping through.

{

//address is removed

address\_list.erase(address\_list.begin()+i);//if its the same then its erased

return true;

}

}

return false;//come out of loop if its not found then return false

}

string AddressBook::returnListOfAddresses()

{

stringstream info;

for (int i=0;i<address\_list.size();i++)//traverse through address\_list vector

{

info <<"Address: "<< endl;

info << address\_list[i]->toString() << endl;//prints all the elements of the vector in a proper format using toString function

}

return info.str();//return as a string datatype

}

string AddressBook::searchAddress(string lastName) //searchAddress function or method with lastName as parameter as entered by user

{

for (int i =0; i<address\_list.size(); i++) //traverse through address\_list vector

{

if(address\_list[i]->getlastName() == lastName)//the current element inside vector gets lastName and checks with the parameter entered by the user to see if they are equal.if they are equal go inside.

{

//name found

//return address

return address\_list[i]->getStreetAddress(); //inside the if statement. current element gets Street Address and returns it

}

}

}

**Q4.**

**Pizza.h**

#include <string>//allows the user of string data type

#include "FoodItem.h"//include FoodItem header file

#pragma once //be included only once

using namespace std;//allows the use of classes objects and functions under a name

class Pizza : public FoodItem //Pizza class inherit from FoodItem class

{

private:

int size;

int number\_of\_toppings;

public:

Pizza(int size, int number\_of\_toppings, int calories, double price, string name);//constructor

int getSize();

int getNumberofToppings();

string toString();

};

**Pizza.cpp**

#include <string>//allows the user of string data type

#include "Pizza.h"//includes Pizza header file

#include <sstream> //allows the use of stringstream data type

using namespace std;//allows the use of classes objects and functions under a name

Pizza::Pizza(int size, int number\_of\_toppings, int calories, double price, string name) : FoodItem(name,calories,price) //Pizza constructor passing name,calories and price to FoodItem class through inheritance

{

this->size = size;//set size to the size in parameter use this keyword because same variable name

this->number\_of\_toppings = number\_of\_toppings;

}

int Pizza::getSize()

{

return size;

}

int Pizza::getNumberofToppings()

{

return number\_of\_toppings;

}

string Pizza::toString()

{

stringstream info;

info << "size: " << size << endl;//put string and size variable into info

info << " Number of Toppings: " << number\_of\_toppings << endl;

return info.str();

}

**Pasta.h**

#include <string>//allows the user of string data type

#include "FoodItem.h"//includes FoodItem header file

#pragma once // #include only once

using namespace std;//allows the use of classes objects and functions under a name

class Pasta : public FoodItem //Class Pasta inheriting from FoodItem

{

private:

string type;

public:

Pasta(string type,string name, int calories, double price);//constructor

string getType();

string toString();

};

**Pasta.cpp**

#include <string>//allows the user of string data type

#include "Pasta.h"//include Pasta header file

#include <sstream>//allows the use of stringstream data type

using namespace std;//allows the use of classes objects and functions under a name

Pasta::Pasta(string type,string name, int calories, double price): FoodItem(name,calories,price)//pasta constructor with pasta passing variables to FoodItem through inheritance

{

this->type = type;//sets variable type to type from parameter

}

string Pasta::getType()

{

return type;

}

string Pasta::toString()

{

stringstream info;

info << type<< endl;//puts type into info

return info.str();//converts info to string

}

**FoodItem.h**

#pragma once//only get included once

#include <string>//allow use of string datatype

using namespace std;//allows the use of classes objects and functions under a name

class FoodItem//FoodItem class format

{

private://the variables in this class can only be accessed by this class.

string name;

int calories;

double price;//double because price might have decimal places

public://any class can call the functions in this class.

FoodItem(string name, int calories, double price);//constructor

string getName();

int getCalories();

double getPrice();

string toString();

};

**FoodItem.cpp**

#include <string>//allows the user of string data type

#include "FoodItem.h" //include FoodItem header file

#include <sstream> //allows the use of stringstream data type

using namespace std;//allows the use of classes objects and functions under a name

FoodItem::FoodItem(string name, int calories, double price)//constructor

{

//initialise the variables of the object that will be created

this->name = name; //set name to name from parameter

this->calories = calories;//set calories to calories from parameter

this->price = price;//set price to price from parameter

}

string FoodItem::getName()

{

return name;

}

int FoodItem::getCalories()

{

return calories;

}

double FoodItem::getPrice()

{

return price;

}

string FoodItem::toString()

{

stringstream info;

info << "Name: " << name << endl;//put string and string variable into info

info << "Calories: " << calories << endl;//put string and string variable into info

info << "Price: " << price << endl;//put string and string variable into info

return info.str();//returns info as string

}

**OrderApp.cpp**

#include <string> //allows the user of string data type

#include "Pasta.h" //include Pasta header file

#include "Pizza.h"//include Pizza header file

#include <iostream>//allows the use of input and output

#include <vector>//allows the use of vector

#include <fstream>//allows the use of ifstream and ofstream.writes and reads textfile

int option = 0;//initialise and set variables

int size;

int number\_of\_toppings;

string type;

vector <FoodItem\*> orderList;//vector orderList of FoodItem as datatype

void option1() //void because there is no return self declared function so that it is more organised

{

cout <<" Order Pizza Selected" << endl;//output string to ask the user

cout << " What size would you like?" << endl;

cin >> size;//gets user input and puts it in size variable

cout << "The Number of Toppings?" << endl;

cin >> number\_of\_toppings;

//created object

Pizza \* pizzaObj = new Pizza(size, number\_of\_toppings, 800, 24.90, "ShawnsPizza");//Pizza Object created with calories,price and name preset.

//add to order

orderList.push\_back(pizzaObj);//push the pizza Object into the orderList vector

cout <<"Pizza added to Order List" << endl;

}

void option2()

{

cout <<" Order Pasta Selected" << endl;

cout << " What type would you like?" << endl;

cin >> type;

//created object

Pasta \* pastaObj = new Pasta(type,"ShawnsPasta",350,15.90);//Pasta Object created with calories,price and name preset.

//add to order

orderList.push\_back(pastaObj);//push pasta Object that was created into the orderList vector

cout <<"Pasta added to Order List" << endl;

}

void option3()

{

cout <<" Print current order Selected" << endl;

for (int i=0; i < orderList.size(); i++)//traverse through orderList vector

{

cout << orderList[i]->toString() << endl;//current element points to toString function then output nicely;

ofstream outfile("order.txt");//sets the order.txt file for recording the messages.creates a txt file if does not exist.

outfile << orderList[i]->toString()<<endl;//current element of the vector points to toString which will output nicely then outputs in the txt file.

}

}

void option4()

{

cout <<"Compute delivery time Selected" << endl;

int time = 20;

for (int i=0; i < orderList.size(); i++) //traverse orderList vector and check for the pizzas and pastas inside then add time accordingly

{

if (orderList[i]->getName() == "ShawnsPizza")//current element gets Name and checks if its same as "ShawnsPizza"

{

time += 7;//if it is then adds 7 to the time variable.

}

else if (orderList[i]->getName() == "ShawnsPasta")//current element gets Name and checks if its same as "ShawnsPasta"

{

time += 10;//if it is then adds 10 to the time variable.

}

}

cout << "Approximate delivery time: " << time << "mins" << endl; // total time will be printed

}

void option5()

{

double cost = 0;

for (int i=0; i < orderList.size(); i++)//traverse orderList vector

{

if (orderList[i]->getName() == "ShawnsPizza")//if the Name of the current element is equal to "ShawnsPizza"

{

cost += 24.90;//add 24.90 to cost variable

}

else if (orderList[i]->getName() == "ShawnsPasta")//if the name of the current element is equal to "ShawnsPasta"

{

cost += 15.90;//add 15.90 to cost variable

}

}

double total = cost+(cost\*0.1)+5;//total variable = cost which is the total cost of the items in the orderList + 10% of it then + 5

cout << "Total cost : " << total << endl;

}

using namespace std; //allows the use of classes objects and functions under a name

int main()

{

cout << "Order Menu" << endl;

cout << "1. Order a pizza" << endl;

cout << "2. Order a pasta" << endl;

cout << "3. Print current order" << endl;

cout << "4. Compute delivery time" << endl;

cout << "5. Print total cost" << endl;

cout << "6. Quit" << endl;

cin>>option; //user input put in option variable

if (option == 1)

{

option1();//self declared function

main();//returns to menu

}

else if (option == 2)

{

option2();

main();

}

else if (option == 3)

{

option3();

main();

}

else if (option == 4)

{

option4();

main();

}

else if (option == 5)

{

option5();

main();

}

}