Q1:

#include<iostream>

using namespace std;

int main(){

float totalcost, totalsalary, yearlyrent, electricitycost,markedupprice,listcost;

float markeduppercentage;

cout << "Please enter the total cost of merchandise: ";

cin >> listcost;

cout << endl;

cout << "Please enter the yearly rent: ";

cin >> yearlyrent;

cout << endl;

cout << "Please enter the salary of the employees: ";

cin >> totalsalary;

cout << endl;

cout << "Please enter the elctricity cost: ";

cin >> electricitycost;

cout << endl;

totalcost = listcost + yearlyrent + totalsalary + electricitycost;

//marked up price is 110 percent of the total cost

markedupprice = 1.1 \* totalcost;

//list price is inclusive of the 15% discount i.e. list price is 85% of the markedupprice

listcost = 0.85 \* markedupprice;

cout << "Marked up price: " << markedupprice << endl;

cout << "List price (sale price): " << listcost << endl;

return 0;

}

---

Please enter the total cost of merchandise: 2000

Please enter the yearly rent: 300

Please enter the salary of the employees: 500

Please enter the elctricity cost: 200

Marked up price: 3300

List price (sale price): 2805

----

Q2:

#include <iostream>

#include <iomanip>

using namespace std;

//Named constants

const double COST\_OF\_SMALL\_CUP\_ = 1.75;

const double COST\_OF\_MEDIUM\_CUP = 1.90;

const double COST\_OF\_LARGE\_CUP = 2.00;

const int SIZE\_OF\_SMALL\_CUP = 9;

const int SIZE\_OF\_MEDIUM\_CUP = 12;

const int SIZE\_OF\_LARGE\_CUP = 15;

//Function declarations

void printMenu();

void sellCoffe(int &smallSold, int &mediumSold, int &largeSold, int &coffeSold, double &moneyMade);

void cupsSold(int small, int medium, int large);

void coffeSold(int amountCoffe);

void moneyMade(double money);

int main()

{

int choice;

int smallSold = 0, mediumSold = 0, largeSold = 0;

int coffe = 0;

double money = 0;

bool doExit = false;

cout << "\*\*Welcome to the coffe shop\*\*" << endl;

//Menu

while (!doExit)

{

//Display the menu

printMenu();

//Prompt the user to enter the choice

cout << "Enter your choice: ";

cin >> choice;

switch (choice)

{

case 1:

//Sell the coffee

sellCoffe(smallSold, mediumSold, largeSold, coffe, money);

break;

case 2:

//print number of cups of each size sold

cupsSold(smallSold, mediumSold, largeSold);

break;

case 3:

//print the amount of coffe sold

coffeSold(coffe);

break;

case 4:

//print the money made

moneyMade(money);

break;

case 5:

//Set the boolean value to true, to exit

doExit = true;

break;

default:

cout << "\*\*Wrong choice" << endl;

}

}

system("pause");

return 0;

}

//Prints the menu that shows the user how to use the program

void printMenu()

{

cout << "\*\*\*\*\*Menu\*\*\*\*\*" << endl;

cout << "1. Buy a coffe" << endl;

cout << "2. Show the totoal number of cups sold" << endl;

cout << "3. Show the total amound of coffe sold" << endl;

cout << "4. Show the total money made." << endl;

cout << "5. Leave the menu" << endl;

}

//Sells coffee

void sellCoffe(int &smallSold, int &mediumSold, int &largeSold, int &coffeSold, double &moneyMade)

{

int size = 0, cups = 0;

while (true)

{

cout << "1. Samll coffe cup(9oz) " << endl;

cout << "2. Medium coffe cup(12oz) " << endl;

cout << "3. Large coffe cup(15oz) " << endl;

cout << "4. Order finished " << endl;

cout << "Enter your choice of coffe cup: ";

cin >> size;

if (size == 4)

{

break;

}

cout << "Enter the number of cups: ";

cin >> cups;

if (size == 1)

{

smallSold = smallSold + cups;

coffeSold = coffeSold + cups \* 9;

moneyMade = moneyMade + (cups \* 1.75);

}

else if (size == 2)

{

mediumSold = mediumSold + cups;

coffeSold = coffeSold + cups \* 12;

moneyMade = moneyMade + (cups \* 1.90);

}

else if (size == 3)

{

largeSold = largeSold + cups;

coffeSold = coffeSold + cups \* 15;

moneyMade = moneyMade + (cups \* 2);

}

else

cout << "Sorry! wrong choices. try again!" << endl;

}

}

//Prints the number of cups of each size sold

void cupsSold(int small, int medium, int large)

{

cout << setprecision(0) << fixed << "Total number of small cups sold: " << small << endl;

cout << "Total number of medium cups sold: " << medium << endl;

cout << "Total number of large cups sold: " << large << endl;

}

//Prints the total amount of coffee sold

void coffeSold(int amountCoffe)

{

cout << setprecision(0) << fixed << "Total amount of coffe sold: " << amountCoffe << " oz" << endl;

}

//Prints the total money made

void moneyMade(double money)

{

cout << setprecision(2) << fixed << "Total money made: $" << money << endl;

}

---

\*\*Welcome to the coffe shop\*\*

\*\*\*\*\*Menu\*\*\*\*\*

1. Buy a coffe

2. Show the totoal number of cups sold

3. Show the total amound of coffe sold

4. Show the total money made.

5. Leave the menu

Enter your choice: 1

1. Samll coffe cup(9oz)

2. Medium coffe cup(12oz)

3. Large coffe cup(15oz)

4. Order finished

Enter your choice of coffe cup: 2

Enter the number of cups: 2

1. Samll coffe cup(9oz)

2. Medium coffe cup(12oz)

3. Large coffe cup(15oz)

4. Order finished

Enter your choice of coffe cup: 3

Enter the number of cups: 1

1. Samll coffe cup(9oz)

2. Medium coffe cup(12oz)

3. Large coffe cup(15oz)

4. Order finished

Enter your choice of coffe cup: 2

Enter the number of cups: 2

1. Samll coffe cup(9oz)

2. Medium coffe cup(12oz)

3. Large coffe cup(15oz)

4. Order finished

Enter your choice of coffe cup: 4

\*\*\*\*\*Menu\*\*\*\*\*

1. Buy a coffe

2. Show the totoal number of cups sold

3. Show the total amound of coffe sold

4. Show the total money made.

5. Leave the menu

Enter your choice: 2

Total number of small cups sold: 0

Total number of medium cups sold: 4

Total number of large cups sold: 1

\*\*\*\*\*Menu\*\*\*\*\*

1. Buy a coffe

2. Show the totoal number of cups sold

3. Show the total amound of coffe sold

4. Show the total money made.

5. Leave the menu

Enter your choice: 3

Total amount of coffe sold: 63 oz

\*\*\*\*\*Menu\*\*\*\*\*

1. Buy a coffe

2. Show the totoal number of cups sold

3. Show the total amound of coffe sold

4. Show the total money made.

5. Leave the menu

Enter your choice: 4

Total money made: $9.60

\*\*\*\*\*Menu\*\*\*\*\*

1. Buy a coffe

2. Show the totoal number of cups sold

3. Show the total amound of coffe sold

4. Show the total money made.

5. Leave the menu

Enter your choice: 5

---

Q3:

a)

#include<iostream>

#include<cstdlib>

using namespace std;

// declaring the node

struct node

{

int data;

struct node \*link;

}\*front, \*rear;

// declaring the queue class

class Queue

{

public:

void insert(int);

void display();

void del();

// constructor to initialize the front and rear of the node

Queue()

{

front = NULL;

rear = NULL;

}

};

// main function

int main()

{

int choice, item;

// creating an object of the Queue class

Queue ql;

// while loop which continues till the user opts out

while (1)

{

// defining the user interface

cout<<"\n----------------"<<endl;

cout<<"Queue Operations";

cout<<"\n----------------"<<endl;

cout<<"1.Insert Element into the Queue"<<endl;

cout<<"2.Delete Element from the Queue"<<endl;

cout<<"3.Traverse the Queue"<<endl;

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

cout<<"Enter your Choice: ";

// taking input of the choice

cin>>choice;

// switch case

switch(choice)

{

// insert element into the queue

case 1:

cout<<"Enter value to be inserted into the queue: ";

cin>>item;

ql.insert(item);

break;

// delete element from the queue

case 2:

ql.del();

break;

// print the elements of the queue

case 3:

ql.display();

break;

// break out of the loop

case 4:

exit(1);

break;

// any other response

default:

cout<<"Wrong Choice"<<endl;

}

}

return 0;

}

// function to insert an element into the queue

void Queue::insert(int item)

{

node \*tmp;

tmp = new (struct node);

tmp->data = item;

tmp->link = NULL;

if (front == NULL)

front = tmp;

else

rear->link = tmp;

rear = tmp;

}

// function to delete an element from the queue

void Queue::del()

{

node \*tmp;

// if the queue has no elements

if (front == NULL)

cout<<"Queue Underflow"<<endl;

else

{

tmp = front;

cout<<"Element Deleted: "<<tmp->data<<endl;

front = front->link;

free(tmp);

}

}

// function to traverse the queue and print the elements in the queue

void Queue::display()

{

node \*ptr;

ptr = front;

// if the queue is empty

if (front == NULL)

cout<<"The Queue is empty"<<endl;

else

{

cout<<"The Queue elements :"<<endl;

while (ptr != NULL)

{

cout<<ptr->data<<" ";

ptr = ptr->link;

}

cout<<endl;

}

}

--

----------------

Queue Operations

----------------

1.Insert Element into the Queue

2.Delete Element from the Queue

3.Traverse the Queue

4.Quit

Enter your Choice: 1

Enter value to be inserted into the queue: 5

----------------

Queue Operations

----------------

1.Insert Element into the Queue

2.Delete Element from the Queue

3.Traverse the Queue

4.Quit

Enter your Choice: 1

Enter value to be inserted into the queue: 12

----------------

Queue Operations

----------------

1.Insert Element into the Queue

2.Delete Element from the Queue

3.Traverse the Queue

4.Quit

Enter your Choice: 1

Enter value to be inserted into the queue: 16

----------------

Queue Operations

----------------

1.Insert Element into the Queue

2.Delete Element from the Queue

3.Traverse the Queue

4.Quit

Enter your Choice: 1

Enter value to be inserted into the queue: 25

----------------

Queue Operations

----------------

1.Insert Element into the Queue

2.Delete Element from the Queue

3.Traverse the Queue

4.Quit

Enter your Choice: 3

The Queue elements :

5 12 16 25

----------------

Queue Operations

----------------

1.Insert Element into the Queue

2.Delete Element from the Queue

3.Traverse the Queue

4.Quit

Enter your Choice: 2

Element Deleted: 5

----------------

Queue Operations

----------------

1.Insert Element into the Queue

2.Delete Element from the Queue

3.Traverse the Queue

4.Quit

Enter your Choice: 5

Wrong Choice

----------------

Queue Operations

----------------

1.Insert Element into the Queue

2.Delete Element from the Queue

3.Traverse the Queue

4.Quit

Enter your Choice: 3

The Queue elements :

12 16 25

----------------

Queue Operations

----------------

1.Insert Element into the Queue

2.Delete Element from the Queue

3.Traverse the Queue

4.Quit

Enter your Choice: 1

Enter value to be inserted into the queue: 30

----------------

Queue Operations

----------------

1.Insert Element into the Queue

2.Delete Element from the Queue

3.Traverse the Queue

4.Quit

Enter your Choice: 3

The Queue elements :

12 16 25 30

----------------

Queue Operations

----------------

1.Insert Element into the Queue

2.Delete Element from the Queue

3.Traverse the Queue

4.Quit

Enter your Choice: 4

-----

b)

1) Array is a collection of elements of similar data type. Linked list is an ordered collection of elements of same type, which are connected to each other using pointers.

2)Elements can be accessed directly by using their index in array. For example, arr[0]

Is the first element and arr[3] is the 4th element. However, we need to sequentially traverse the whole linked list to access the element/node in a linked list. Linked list supports sequential access whereas array supports random access.

3) Because array’s size is fixed, insertion and deletion operations in array are slower. In linked list, we can traverse the whole list for getting nth element. Every element has a link to its previous element. That’s why insertion and deletion operations are faster in linked list.

Q4:

#include <iostream>

#include<stack>

using namespace std;

int main()

{

stack<string> stack;

//The list of students names are: Felicity, Lily, Amy, Sally, Tina

stack.push("Felicty");

stack.push("Lily");

stack.push("Amy");

stack.push("Sally");

stack.push("Tina");

while (!stack.empty()) {

cout<< stack.top()<<" ";

stack.pop();

}

return 0;

}

---

Tina Sally Amy Lily Felicty %

---

I use STL stack to store a list of student names and display in the reverse order. Stack works on “last in first out” rule. Elements that go last will come first from stack.