Question 1: TASK 6

```
#include <iostream>
#include <string>

using namespace std;

class Node {
public:
    string item_name;
    int price;
    Node* next;

    Node(string name, int price)
    {
        item_name = name;
        price = price;
        next = NULL;
    }
};

class Queue {
private:
    Node* front;
    Node* rear;
    int length;

public:
    Queue()
     {
        front=NULL;
        rear=NULL;
        length=0;
     }

    void enqueue(string value, int price) {
        Node* new_node = new Node(value, price);
        if (rear == nullptr)
         {
            front = new_node;
            rear = new_node;
        } else
        {
            rear->next = new_node;
            rear = new_node;
        }
        length++;
        cout<<"\n";
        cout<<"*********************************\n";
        cout << value << " added to the queue.\n";
        cout<<"*********************************\n";
```

```
    }


    void dequeue() {
        if (front == nullptr)
        {
            cout << "\nThe queue is empty! No orders to process.\n";
            return;
        }

        Node* curr = front;
        front = front->next;
        cout<<"\n*********************************\n";

        cout << "Processing order: " << curr->item_name << " ($" << curr->price << ")\n";
        cout<<"\n*********************************\n";

        delete curr;
        length--;

        if (front == nullptr)
        {
            rear = nullptr;
        }
    }

    // Display current orders in the queue
    void display() {
        if (front == nullptr)
         {
            cout << "\nNo current orders in the queue.\n";
            return;
        }

        cout<<"\n*********************************\n";
        cout << "Current orders in the queue:\n";
        Node* curr = front;
        while (curr != nullptr)
        {
            cout << "- " << curr->item_name << " ($" << curr->price << ")\n";
            curr = curr->next;
        }
        cout<<"\n*********************************\n";

    }

    // Check if the queue is empty
    bool isEmpty() {
        return front == nullptr;
    }
};
```

```
void displayMenu()
{
   cout << "\nAvailable food items and beverages:\n";
   cout << "1. Sabzi - $120\n";
   cout << "2. Bendi - $150\n";
   cout << "3. Biryani - $250\n";
   cout << "4. Remove the proceed orders \n";
   cout<<  "5. Display the orders in Queue\n";
   cout << "6. Quit\n";
}

int main() {
   Queue orderQueue;
   int choice;

   while (1)
   {
      displayMenu();
      cout << "Select an option (1-4): ";
      cin >> choice;

      switch (choice) {
         case 1:
            orderQueue.enqueue("Sabzi", 120);
            break;
         case 2:
            orderQueue.enqueue("Bendi", 150);
            break;
         case 3:
            orderQueue.enqueue("Biryani", 250);
            break;
         case 4:
            orderQueue.dequeue();
            break;

         case 5:
            orderQueue.display();
            break;

         case 6:
            cout << "Exiting the application.\n";
            return 0;
         default:
            cout << "Enter a valid choice.\n";
            continue;
      }
   }


   return 0;
}
```

Outputs ScreenShots:

1)

```
Available food items and beverages:
1. Sabzi - $120
2. Bendi - $150
3. Biryani - $250
4. Remove the proceed orders
5. Display the orders in Queue
6. Quit
Select an option (1-4): 1

**********************************
Sabzi added to the queue.
**********************************

Available food items and beverages:
1. Sabzi - $120
2. Bendi - $150
3. Biryani - $250
4. Remove the proceed orders
5. Display the orders in Queue
6. Quit
Select an option (1-4): 2

**********************************
Bendi added to the queue.
**********************************

Available food items and beverages:
1. Sabzi - $120
2. Bendi - $150
3. Biryani - $250
4. Remove the proceed orders
5. Display the orders in Queue
6. Quit
Select an option (1-4): 3

**********************************
Biryani added to the queue.
**********************************
```

2)

```
1. Sabzi - $120
2. Bendi - $150
3. Biryani - $250
4. Remove the proceed orders
5. Display the orders in Queue
6. Quit
Select an option (1-4): 5

**********************************
Current orders in the queue:
- Sabzi ($0)
- Bendi ($0)
- Biryani ($0)

**********************************

Available food items and beverages:
1. Sabzi - $120
2. Bendi - $150
3. Biryani - $250
4. Remove the proceed orders
5. Display the orders in Queue
6. Quit
Select an option (1-4): 4

**********************************
Processing order: Sabzi ($0)

**********************************

Available food items and beverages:
1. Sabzi - $120
2. Bendi - $150
3. Biryani - $250
4. Remove the proceed orders
5. Display the orders in Queue
6. Quit
Select an option (1-4): 5

**********************************
Current orders in the queue:
- Bendi ($0)
- Biryani ($0)

**********************************

Available food items and beverages:
1. Sabzi - $120
2. Bendi - $150
3. Biryani - $250
4. Remove the proceed orders
5. Display the orders in Queue
6. Quit
Select an option (1-4):
```

Question 2:

```
#include<iostream>


using namespace std;

class Nodes
{
   public:
   int info;
   Nodes *Next; //It will point to the object of the class made in the main function

   Nodes(int value)
   {
      info= value;
      Next=NULL;

   }

};


class Queue
{
   private:
         Nodes *front; //Like a head
         Nodes *rear;
         int capacity;
         int length;

   public:
      Queue(int size)
      {
         capacity=size;
         length=0;
         front=NULL;
         rear=NULL;
      }

   void enqueue(int value)
   {

      Nodes *new_node= new Nodes(value);
      if(rear==NULL)
      {
         front=new_node;
         rear=new_node;

      }
      else
```

```
    {
        rear->Next=new_node;
        rear=new_node;
    }
    length++;

}

int dequeue()
{
    if(front==NULL)
    {
        cout<<"THe queue is empty!!\n";
        return -1;
    }

    Nodes *curr=front;
    front=front->Next;
    int value=curr->info;
    delete curr;
    length--;
    return value;
}




void display()
{
    Nodes *curr=front;
    for(int i=0;i<length;i++)
    {
        cout<<curr->info;
        curr=curr->Next;
    }
}
 void insert_by_position(int value, int position)
{
    if(position<1 || position>length+1)
    {
        cout<<"\nInvalid position input\n";
        return;
    }

    Nodes *n= new Nodes(value);

    if(position==1)
    {
        n->Next=front;
        front=n;
    }
    else //if the input position is between 1 and the length of the linked list
```

```
    {
        Nodes *Curr_ptr=front;

        for( int i=1; i<position-1;i++)
        {
            Curr_ptr=Curr_ptr->Next;
        }
        n->Next=Curr_ptr->Next;
        Curr_ptr->Next=n;
    }
    length++;
}

void X_time_duplicating()
{
    Nodes *Curr=front;
    int original=length;
    int position=1;
    int counter=1;
    for( int i=1;i<=original;i++)
    {
        int value=Curr->info;

    if(value > 1)
    {

        int times=Curr->info;
        if(Curr->Next !=NULL)
        Curr=Curr->Next;
        counter=1;
        while (times!=1)
        {
            insert_by_position(value,position);
            counter+=1;
            times-=1;
        }
        position+=counter;

    }
    else
        {
            if(Curr->Next !=NULL)
                Curr=Curr->Next;
            position+=1;

        }


}
}
};
```

```
int main(void)
{
   Queue Q_LinkedLIst(5);
   Q_LinkedLIst.enqueue(1);
   Q_LinkedLIst.enqueue(2);
   Q_LinkedLIst.enqueue(3);
   Q_LinkedLIst.enqueue(4);
   Q_LinkedLIst.enqueue(1);
   Q_LinkedLIst.enqueue(6);
   Q_LinkedLIst.enqueue(7);
   Q_LinkedLIst.enqueue(8);
   Q_LinkedLIst.enqueue(9);

cout<<"\nbefore:\n";


   Q_LinkedLIst.display();

   Q_LinkedLIst.X_time_duplicating();

cout<<"\nAfter:\n";
   Q_LinkedLIst.display();


   return 0;
}
```