Dsa-Lab-Task : 7

Roll No: 24P-0706 **Dept: BS-CS**

Name: Aazan Noor Khuwaja **Section : 3D**

Qns1:

```
#include<iostream>
using namespace std;
class node{
   public:
   int id;
   string name;
   float price;
   node *next;
   node(int i,string n,float f):id(i),price(f),name(n){
      next=nullptr;
   }
};
class menu
{
   public:
   node *head;
   menu()
   {
      head=nullptr;
   }
```

```
bool is_empty()
{
   return (head==nullptr);
}
void insert(int in_id,string in_na,float in_pr)
{
   node *n_node=new node(in_id,in_na,in_pr);
   if(is_empty())
   {
      head=n_node;
      return;
   }
   node*temp=head;
   while(temp->next!=nullptr)
   {
      temp=temp->next;
   }
   temp->next=n_node;
}

node* find_that_order()
   {
      int search_id;
      cout <<"Enter the id of the item to order: "<<endl;
      cin>>search_id;

      node *temp=head;
```

```
        while(temp!=nullptr && temp->id!=search_id){
            temp=temp->next;
        }
        return temp;
    }


    void display_menu(){
    if (is_empty()){
        cout<<"menu is empty.\n";
        return;
    }
    node*temp=head;
    cout<<"---- cafeteria Menu ----\n";
    while(temp!=nullptr){
        cout<<"ID: "<<temp->id<<". \nName:"<<temp->name<<" \nRs. "<<temp->price<<endl;
        cout<<"-------------------------"<<endl;
        temp=temp->next;
    }

}

};

class order_node{
public:
    int id;
```

```
    string name;
    float price;
    order_node *next;

    order_node(int i,string n,float f):id(i),price(f),name(n){
        next=nullptr;
    }
};

class queue{
    public:
    order_node*front,*rear;
    queue()
    {
        front=rear=nullptr;
    }
    bool is_empty_queue()
    {
        return (front==nullptr);
    }
    void inseart(order_node* new_order)
    {
        if(is_empty_queue()){
            front=rear=new_order;
            return;
        }
        rear->next=new_order;
```

```
    rear=new_order;
  }

  void out()
  {
    if(is_empty_queue()){
    cout <<"No Orders to process!"<<endl;
    return;
    }
    order_node*temp=front;
    cout<<"processing order ID: "<<temp->id<<" -> "<<temp->name<<" recieve your Order!
\n";
    front=front->next;
    delete temp;
    if(!front)
    {
      rear=nullptr;
    }
  }

  void show_order()
  {
    if(is_empty_queue())
    {
      cout<<"No order in waiting!"<<endl;
      return ;
    }
```

```
        order_node*temp=front;
        while(temp!=nullptr){
        cout<<"ID: "<<temp->id<<"\nName: "<<temp->name<<"\nRs: "<<temp->price<<endl;
        temp=temp->next;
        }
    }

};


int main(){
    menu d;
    d.insert(1,"Chiken Biryani",250);
    d.insert(2,"Fries",120);
    d.insert(3,"Half Biryani",180.9);
    d.insert(4,"Nashta",150.4);
    d.insert(5,"Shake",300.50);
    d.insert(6,"Daal",100);
    d.insert(7,"Tea",80);
    d.insert(8,"Cold Drink",90);
    d.insert(9,"Lobya",130);
    d.insert(10,"Roti",20);

    queue q_cafetaria;
    int ch;
      while(true){
        cout<< "\n======Cafeteria-System======\n";
```

```
    cout<< "1.Display Menu\n"<<"2.place order\n"<<"3.show pending ordrs\n"<< "4.Process next Order\n"<< "5.Exit\n";
    cout<< "Pick your choice: ";
    cin>>ch;
    switch (ch)
    {
    case 1:
      d.display_menu();
      break;
    case 2:{
      node* order=d.find_that_order();
      if(order!=nullptr)
      {
        order_node* n_order=new order_node(order->id,order->name,order->price);
        q_cafetaria.inseart(n_order);
        cout <<"Your Order placed with this id:" <<order->id<<endl;
      }
      else {
        cout <<"Not a valid ID!\n Try again!"<<endl;
      }
    }
      break;
    case 3:
      q_cafetaria.show_order();
      break;
    case 4:
      q_cafetaria.out();
```

```
            break;
        case 5:
        cout <<"Leaving Cafeteria System ..."<<endl;
        return 0;
            break;
        default:
            cout<<"Not a valid choice so try again !"<<endl;
            break;
        }
    }

}
```

Qns 2:

```
#include<iostream>
#include<string>
using namespace std;
class queue_circular{
    public:
    string *arr;
    int cap, siz_cap;
    int f,r;
    string name ;
    queue_circular()
    {
        cap=5;
```

```
        arr=new string[cap];
        f=r=-1;
        siz_cap=0;
    }
    bool is_full(){
    return (((r+1)%cap == f));
}
    bool is_empty()
    {
        return (f==-1);
    }
    void add_patient(string n)
    {
        if(is_full())
        {
            cout <<"waiting is already FUll! \n Can't add more!"<<endl;
            return;
        }
        if(is_empty())
        {
            f=r=0;
        }
        else {
            r=(r+1)%cap;
        }
        arr[r]=n;
        siz_cap++;
        cout <<"Patient: "<<n<<" Appointment Given!"<<endl;
    }
```

```
void send_to_doctor()
{
   if(is_empty())
   {
      cout <<"No patients in waiting !"<<endl;
   }
   else {
      cout<<"sending "<<arr[f]<<" to Docter \nSend Next Patient!"<<endl;
   }
   if(f==r)
   {
      f=r=-1;
   }
   else{
      f=(f+1)%cap;
   }
   siz_cap--;
}

void next_patient()
{
   if(is_empty())
   {
      cout <<"No patient Waiting !"<<endl;
   }
   else{
      cout<<"Next Patient is :"<<arr[f]<<endl;
   }
}
```

```
    void display_patients()
    {
        if(is_empty()){
            cout<<"No patients currently waiting"<<endl;;
            return;
        }
        cout <<"Total Waiting patients are:"<<endl;
        int i=f;
        while(true)
        {
            cout <<"Patient: "<<arr[i]<<endl;
            if(i==r)break;
            i=(i+1)%cap;
        }
    }

};
int main()
{
    queue_circular q;
    int c;
    string name;
    while(true)
    {
        cout<<"***Doctor System*** \n1.Add patient\n"<<"2.Send next patient to doctor\n"<<"3.See
who is next\n4.Exit"<<endl;
        cin>>c;
        switch (c)
```

```
        {
        case 1:{
        cout <<"Enter the name of patient:"<<endl;
        cin.ignore();
        getline(cin,name);
        q.add_patient(name);
    }
        break;
        case 2:
        q.send_to_doctor();
        break;
        case 3:
        q.next_patient();
        break;
        case 4:
        return 0;
        break;
        default:
        cout <<"Not a valid Choice!\n try again!"<<endl;
           break;
        }
    }
}
```