UNIVERSITY OF THE PUNJAB
(Gujranwala Campus)

HOMEWORKS, LABS, ASSIGNMENTS
HOTEL MANAGEMENT SYSTEM REPORT
DATA STRUCTURE AND ALGORITHMS

PREPARED BY:

LAIQA GHAFFAR (BIT21207)

MEHLAB SHAHBAZ (BIT21217)

ESHA IFTIKHAR (BIT21227)

UNAISAH NAJAM (BIT21246)

SUBMITTED TO:

PROF. SALMAN NASEER

TABLE OF CONTENTS PAGE NO

ACKNOWLEDGEMENT

First and fore most we thank our teacher ***Prof. Salman Naseer*** *has assigned us this term paper to bring out our creative capabilities.*

We express our gratitude to our ***parents*** *for being a continuous source of encouragement for all their financial aid.*

We would like to acknowledge the assistance provided to us by the library staff of ***LOVELY PROFESSIONAL UNIVERSITY****.*

We heartfelt gratitude to our ***class-mates*** *and for helping us to complete our work in time.*

Abstract

This project proposes that the ***efficiency of hotel organizations*** *could be improved by integrating service-oriented operations with project management principles. Such operations would incorporate innovation, proactive attitudes and regulated risk-taking needed to pursue ongoing improvement and proactive response to change in hotel management system.*

Hotel Management project *provides room booking, staff management and other necessary hotel management features. The system allows the manager to post available rooms in the system. Customers can view and book room online.*

Admin *has the power of either approving or disapproving the customer's booking request.*

Other hotel services can also be viewed by ***the customers*** *and can book them too. The system is hence useful for both customers and managers to portably manage the hotel activities.*

HOMEWORKS

HOMEWORK 1:

Question: Design a C++ program to implement a Date class that can perform basic operations such as setting and getting the date and checking for leap years

Code:

```cpp
#include<iostream>
#include<string.h>
using namespace std;
int limit[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
string
name[13]={"Null","January","February","March","April","May","June","July","August
","September","October","November","December"};
class date
{
    private:
        int day;
        int month;
        int year;
    public:
        date();
        date(int ,int ,int );
        void setDate(int ,int ,int);
        void setDay(int);
        void setMonth(int);
        void setYear(int);
        int getDay();
        int getMonth();
        int getYear();
        void IsLeap(int);
        void IsLeap();
        void printDate();
        void printAge();
        void operator -=(date i)
        {
            if(day>i.day)
            day=day-i.day;
            else
            day=i.day-day;
            if(month>i.month)
            month=month-i.month;
            else
            month=i.month-month;
            year=i.year-year;
        }
};
date::date()
{
    day=0;
    month=0;
    year=0;
}
date::date(int d,int m,int y)
{
    this->setDate(d,m,y);
}
void date::setDate(int d,int m,int y)
{
    this->setDay(d);
    this->setMonth(m);
    this->setYear(y);
}
void date::setDay(int d){
    day=((d>=1&&d<=limit[this->getMonth()])?d:1);
    if(this->getMonth()==2&&this->getYear()%4==0);
    day=((d>=1&&d<=29)?d:1);
}
void date::setMonth(int m){
        month=((m>=1&&m<=12)?m:1);
}
void date::setYear(int y){
        year=((y>=1990&&y<=2023)?y:1);
}
int date::getDay()
{
    return day;
}
int date::getMonth()
{
    return month;
}
int date::getYear()
{
    return year;
}
void date::IsLeap()
{
    if(year%4==0)
    cout<<"\tLeap Year"<<endl;
    else
    cout<<"\tNO Leap Year"<<endl;
}
void date::IsLeap(int y)
{
    if(y%4==0)
    cout<<"\tLeap Year"<<endl;
    else
    cout<<"\tNO Leap Year"<<endl;
}
void date::printDate()
{
    cout<<"Date-of-Birth : "<<this->getDay()<<"-"<<name[this->month]<<"-"<<this-
>getYear();
}
void date::printAge()
{
    cout<<"Current   Age : "<<this->getYear()<<"-"<<this->month<<"-"<<this-
>getDay();
}

int main()
{
    int a,b,c,d,e,f,y;
    cout<<"Enter Date of Birth:"<<endl;
    cin>>a>>b>>c;
    date i;
    i.setDate(a,b,c);
    cout<<endl;
    cout<<"Enter current date: "<<endl;
    cin>>d>>e>>f;
    date j(d,e,f);
    i.printDate();
    i.IsLeap();
    i-=j;
    i.printAge();
    cout<<"Enter year that u want to check is leap or not: ";
    cin>>y;
    i.IsLeap(y);
}
```

HOMEWORK 2:

Question:

1. Implement a singly linked list in C++ with basic operations such as insertion, deletion, and traversal. Design a program that prompts the user to perform these operations interactively.
   Code:

```
#include<iostream>
using namespace std;
class node
{
    private:
            int data;
            node *next;
    public:
            node(int );
            void setData(int);
            int getData();
            void setNext(node *);
            node* getNext();
            void showData();
};
node::node(int d)
{
        setData(d);
        next = NULL;
}
void  node::setData(int d)
{
        data = d;
}
int node::getData()
{
        return data;
}
void node::setNext(node* ptr)
{
        next = ptr;
}
node* node::getNext()
{
    return next;
```

```
}
void node::showData()
{
	cout<<data<<"    ";
}
class list
{
	private:
		node* head;
		node* current;
		int size;
	public:
		list();
		void add(int );
		void showList();
		void remove();
		void inserthead(int);
		void forward();
		void back();
		void start();
};
list::list()
{
	head = NULL;
	current = NULL;
	size =0;
}
void list::add(int d)
{
	node *ptr = new node(d);
	if(size==0)
	{
		head = ptr;
		current = ptr;
	}
	else
	{
		ptr->setNext(current->getNext());
		current->setNext(ptr);
		current=ptr;
	}
	size++;
}
void list::inserthead(int d)
{
    node *ptr = new node(d);
    ptr->setNext(head);
    head=ptr;
    current=ptr;
    size++;
}
void list::showList()
{
        node* ptr = head;
        while(ptr!=NULL)
        {
                ptr->showData();
                ptr = ptr->getNext();
        }
}
void list::remove()
{
    if(size==0){
    cout<<"List is Empty";
    }
    if(head==current)
    {
        head=head->getNext();
    }
    node *temp=current;
    node *ptr=head;
    while(ptr!=NULL)
    {
        if(ptr->getNext()==current)
        {
            break;
        }
        ptr=ptr->getNext();
    }
    ptr->setNext(current->getNext());
    current=current->getNext();
    delete temp;
    size--;
}
/****/

void list::forward()
{
    if(current!=NULL)
    current=current->getNext();
```

```
}
void list::back()
{
    node *ptr=head;
    do
    {
        if(ptr->getNext()==current)
        {
            break;
        }
        ptr=ptr->getNext();
    }while(ptr!=NULL);
}
void list::start()
{
    current=head;
}


int main()
{
        cout<<"hello world"<<endl;
        list lst;
        lst.add(20);
        lst.add(60);
        lst.add(80);
        lst.add(90);
        lst.add(70);
        lst.add(40);
        lst.showList();
        cout<<endl;
//      lst.remove();
//      lst.showList();
//      cout<<endl;
//      lst.remove();
//      lst.showList();
        cout<<endl;
        lst.add(10);
        lst.showList();
        cout<<endl;
//      lst.remove();
//      lst.showList();
        list lst1;
        lst1.add(20);
        lst1.add(60);
        lst1.add(80);
        lst1.add(90);
        lst1.add(70);
        lst1.add(10);
        lst1.showList();
        cout<<endl;
        lst1.inserthead(17);
        lst1.showList();
        cout<<endl;
        lst1.inserthead(27);
        lst1.showList();
        cout<<endl;
        lst1.inserthead(37);
        lst1.showList();
        cout<<endl;
        lst1.remove();
        lst1.showList();
}
```

2. Implement a singly circular linked list in C++ with basic operations such as insertion, deletion, and traversal. Design a program that prompts the user to perform these operations interactively.
   Code:

```
#include<iostream>
using namespace std;
class node
{
    private:
        int data;
        node *next;
    public:
        node(int);
        void setData(int);
        int getData();
        void setNext(node*);
        node* getnext();
        void showData();
};
node::node(int d)
{
    setData(d);
    next=NULL;
```

```cpp
}
void node::setData(int d)
{
    data=d;
}
void node::showData()
{
    cout<<" "<<data;
}
int node::getData()
{
    return data;
}
void node::setNext(node *ptr)
{
    next=ptr;
}
node *node::getnext()
{
    return next;
}

class list
{
    private:
        int size;
        node *head;
        node *current;
    public:
        list();
        void addTail(int);
        void addHead(int);
        void showList();
        int TotalElement();
        int removeTail();
        int removeHead();
        void forward();
        void back();
};
list::list()
{
    size=0;
    head=NULL;
    current=NULL;
};
```

```
void list::addHead(int d)
{
    node *ptr=new node(d);
    ptr->setNext(head);
    current->setNext(ptr);
    head=ptr;
    //current=ptr;
    size++;
}
void list::addTail(int d)
{
    node *ptr=new node(d);
    if(size==0)
    {
        head=ptr;
        current=ptr;
        current->setNext(head);
    }
    else
    {
        ptr->setNext(current->getnext());
        current->setNext(ptr);
        current=ptr;
    }
    size++;
}
void list::showList()
{
    if(size==0)
    {
        cout<<"List is Empty"<<endl;
    }
    else
    {
    node *ptr=head;
    do
    {
        ptr->showData();
        ptr=ptr->getnext();
    }while(ptr!=head);
    }
}
int list::TotalElement()
{
    return size;
```

```
}
int list::removeTail()
{
    if(current==head)
    {
        head=head->getnext();
    }
    node *ptr=head;
    node *tem=current;
    int x=tem->getData();
    do
    {
        if(ptr->getnext()==current)
        {
            break;
        }
        ptr=ptr->getnext();
    }while(ptr!=head);
    ptr->setNext(current->getnext());
    current=current->getnext();
    delete tem;
    size--;
    return x;
}
void list::forward()
{
    if(current!=NULL)
    current=current->getnext();
}
void list::back()
{
    node *ptr=head;
    do
    {
        if(ptr->getnext()==current)
        {
            break;
        }
        ptr=ptr->getnext();
    }while(ptr!=head);
}
int main()
{
    cout<<"Hello World"<<endl;
    list singly;
    singly.showList();
    cout<<"\nTotal Elements = "<<singly.TotalElement()<<endl;
    singly.addTail(5);
    cout<<"\nTotal Elements = "<<singly.TotalElement()<<endl;
    cout<<"List: ";
    singly.showList();
    singly.addTail(10);
    singly.addTail(15);
    cout<<"\nTotal Elements = "<<singly.TotalElement()<<endl;
    cout<<"List: ";
    singly.showList();
    cout<<"\ndelete Elemented = "<<singly.removeTail()<<endl;
    cout<<"Total Elements = "<<singly.TotalElement()<<endl;
    cout<<"List: ";
    singly.showList();
    cout<<"\ndelete Elemented = "<<singly.removeTail()<<endl;
    cout<<"Total Elements = "<<singly.TotalElement()<<endl;
    cout<<"List: ";
    singly.showList();
    singly.addHead(100);
    singly.addHead(200);
    cout<<"\nTotal Elements = "<<singly.TotalElement()<<endl;
    cout<<"List: ";
    singly.showList();
}
```

3. Implement a doubly linked list in C++ with basic operations such as insertion, deletion, and traversal. Design a program that prompts the user to perform these operations interactively.
   Code:

```
#include<iostream>
using namespace std;
class node
{
    private:
        int data;
        node *next;
        node *prev;
    public:
        node(int);
        void setData(int);
        int getData();
        void setNext(node*);
        node* getnext();
        void setPrev(node *);
        node* getPrev();
        void showData();
};
node::node(int d)
{
    setData(d);
    next=NULL;
    prev=NULL;
}
void node::setData(int d)
{
    data=d;
}
void node::showData()
{
    cout<<" "<<data;
}
int node::getData()
{
    return data;
}
void node::setNext(node *ptr)
{
    next=ptr;
}
node *node::getnext()
{
    return next;
}
void node::setPrev(node *ptr)
{
        prev = ptr;
}
node* node::getPrev()
{
        return prev;
}
class list
{
    private:
        int size;
        node *head;
        node *current;
    public:
        list();
        void addTail(int);
        void add(int);
        void addHead(int);
        void showList();
        int TotalElement();
        int removeTail();
        int removeHead();
        void forward();
        void back();
        void start();
        void tail();
};
list::list()
{
    size=0;
    head=NULL;
    current=NULL;
};
void list::addHead(int d)
{
    node *ptr=new node(d);
    ptr->setNext(head);
    ptr->setPrev(head->getPrev());
    head->getPrev()->setNext(ptr);
    head=ptr;
    current=ptr;
    size++;
}
void list::add(int d)
{
    node *ptr=new node(d);
    if(size==0)
    {
        head=ptr;
        current=ptr;
        current->setNext(head);
        current->setPrev(head);
    }
    else
    {
        ptr->setNext(current->getnext());
        ptr->setPrev(current);
        current->getnext()->setPrev(ptr);
        current->setNext(ptr);
        current=ptr;
    }
    size++;
}
void list::addTail(int d)
{
    current=head->getPrev();
    node *ptr=new node(d);
        ptr->setNext(current->getnext());
        ptr->setPrev(current);
        current->getnext()->setPrev(ptr);
        current->setNext(ptr);
        current=ptr;
    size++;
}
void list::showList()
{
    if(size==0)
    {
        cout<<"List is Empty"<<endl;
    }
    else
    {
    node *ptr=head;
    do
    {
        ptr->showData();
        ptr=ptr->getnext();
    }while(ptr!=head);
    }
}
int list::TotalElement()
{
    return size;
}
int list::removeTail()
{
    if(current==head)
    {
        head=head->getnext();
    }
    node *ptr=head;
    node *tem=current;
    int x=tem->getData();
    do
    {
        if(ptr->getnext()==current)
        {
            break;
        }
        ptr=ptr->getnext();
    }while(ptr!=head);
    ptr->setNext(current->getnext());
    ptr->getnext()->setPrev(ptr);
    current=current->getnext();
    current->setPrev(ptr);
    delete tem;
    size--;
    return x;
}
void list::forward()
{
    if(current!=NULL)
    current=current->getnext();
}
void list::back()
{
    node *ptr=head;
    do
    {
        if(ptr->getnext()==current)
        {
            break;
        }
        ptr=ptr->getnext();
    }while(ptr!=head);
}
void list::start()
{
    current=head;
}
void list::tail()
{
    current=head->getPrev();
}

int main()
{
    cout<<"Hello World"<<endl;
    list doubly;
    doubly.showList();
    cout<<"\nTotal Elements = "<<doubly.TotalElement()<<endl;
    doubly.add(5);
    cout<<"\nTotal Elements = "<<doubly.TotalElement()<<endl;
    cout<<"List: ";
    doubly.showList();
    doubly.add(10);
    doubly.add(15);
    doubly.add(20);
    doubly.add(25);
    cout<<"\nTotal Elements = "<<doubly.TotalElement()<<endl;
    cout<<"List: ";
    doubly.showList();
    cout<<"\ndelete Elemented = "<<doubly.removeTail()<<endl;
    cout<<"Total Elements = "<<doubly.TotalElement()<<endl;
    cout<<"List: ";
    doubly.showList();
    doubly.tail();
    cout<<"\ndelete Elemented = "<<doubly.removeTail()<<endl;
    cout<<"Total Elements = "<<doubly.TotalElement()<<endl;
    cout<<"List: ";
    doubly.showList();
    doubly.addHead(1000);
    cout<<"\nTotal Elements = "<<doubly.TotalElement()<<endl;
    cout<<"List: ";
    doubly.showList();
    doubly.addTail(2000);
    cout<<"\nTotal Elements = "<<doubly.TotalElement()<<endl;
    cout<<"List: ";
    doubly.showList();
}
```

HOMEWORK 3:

Question: Implement a list in C++ with basic operations such as add, fnd, delete and insert etc. Design a program that prompts the user to perform these operations interactively.

Code:

```
#include<iostream>
using namespace std;
class list
```

```
{
        private:
            int *ptr;
            int length;
            int size;
            int current;
        public:
            list(int);
            void add(int);
            void printList();
            void next();
            void back();
            void start();
            void tail();
            void copy(list );
            void clearList();
            void findPos(int );
            int getLength();
            int getValAtPos(int );
            ~list();
};
list::list(int l)
{
        if(l>0)
        {
                length = l+1;
                size = 0;
                current = 0;
                ptr = new int[length];
                for(int i=0;i<=length;i++)
                    ptr[i]=-1;
        }
}
void list::add(int val)
{
        if(current == length)
        {
            cout<<"list is full"<<endl;

        }
        else
        {

        ptr[++current]=val;
        size++;
        }
}
void list::printList()
{
        for(int i=1;i<=size;i++)
            cout<<ptr[i]<<"   ";
        cout<<endl;
        if(size==0)
            cout<<"List is empty : "<<endl;
}
void list::next()
{
        if(current == length)
            cout<<"current is alread at end, can't move further"<<endl;
        else
            current++;
}
void list::back()
{
        if(current ==1 )
            cout<<"current is already at start, cant move back more "<<endl;
        else
            current--;
}
void list::start()
{
        current = 1;
}
void list::tail()
{
        current = size;
}
void list::copy(list l)
{
        length = l.length;
        size =  l.size;
        current = l.current;
        ptr = new int [length];
        for(int i = 1;i<=length;i++)
            ptr[i]=l.ptr[i];
}
void list::clearList()
{
        size=0;
        current = 0;
        for(int i=1;i<=length;i++)
            ptr[i]=-1;
}
void list::findPos(int val)

{
        for(int i=1;i<=size;i++)
        {
                if(ptr[i]== val)
                {
                        cout<<"value found at index number : "<<i<<endl;
                        return;
                }
        }
        cout<<"value dose not exist : "<<endl;
}
int list::getLength()
{
        return length;
}
int list::getValAtPos(int pos)
{
        if(pos<=size&&pos>=1)
            return ptr[pos];
        else
            {
                cout<<"invalid position"<<endl;
                return -1;
            }

}
list::~list()
{
        delete []ptr;
}
int main()
{
        cout<<"hello world"<<endl;
        list l1(10);
        l1.add(2);
        l1.add(6);
        l1.add(8);
        l1.add(7);
        l1.add(1);
        l1.printList();
        list l2(11);
        l2.copy(l1);
        l2.add(8);
        l2.printList();
        l2.clearList();
        l2.printList();
        l1.findPos(6);

}
```

HOMEWORK 4:

Question: Design a C++ program to implement a stack using an array. Include functionalities to push, pop, and check the top element of the stack. Ensure that the stack can handle a specifed maximum number of elements to prevent overfoo. Implement error handling for stack underfoo conditions. Design a C++ program to implement a stack using a linked list. Defne a code structure for the linked list and include functionalities to push, pop, and check the top element of the stack. Implement error handling for stack underfoo conditions

CODE:

```
#include<iostream>
using namespace std;
/********************************
        stack with array
********************************/
class Stack
{
    private:
        int *arr;
        int top;
        int length;
    public:
        Stack(int);
        void push(int);
        int pop();
        int getTop();
        bool IsFull();
        bool IsEmpty();
};
```

```
Stack::Stack(int l)
{
    length=l;
    arr=new int[length];
    top=-1;
}
void Stack::push(int d)
{
    if(top==length-1)
    {
        cout<<"Stack is full"<<endl;
    }
    else
    {
    arr[++top]=d;
    }
}
int Stack::pop()
{
    if(top==-1)
    {
        return -1;
    }
    else
    {
    return arr[top--];
    }
}
int Stack::getTop()
{
    if(top==-1)
    return -1;
    else
    return arr[top];
}
bool Stack::IsEmpty()
{
    if(top==-1)
    {
        return true;
    }
    else
    return false;
}
bool Stack::IsFull()
```

```
{
    if(top==length-1)
    {
        return true;
    }
    else
    return false;
}

/********************************
        stack with Linked list
********************************/
class node
{
    private:
        int data;
        node *next;
    public:
        node(int);
        void setData(int);
        int getData();
        void showData();
        void setNext(node*);
        node* getNext();
};
node::node(int d)
{
    setData(d);
    next=NULL;
}
void node::setData(int d)
{
    data=d;
}
int node::getData()
{
    return data;
}
void node::showData()
{
    cout<<data<<"  ";
}
void node::setNext(node *ptr)
{
    next=ptr;
```

```
}
node* node::getNext()
{
    return next;
}
class stack
{
    private:
        node *top;
    public:
        stack();
        void push(int);
        int pop();
        int getTop();
        bool IsEmpty();
};
stack::stack()
{
    top=NULL;
}
void stack::push(int d)
{
    node *ptr=new node(d);
    ptr->setNext(top);
    top=ptr;
}
int stack::pop()
{
    int x=top->getData();
    node *temp=top;
    top=top->getNext();
    delete temp;
    return x;
}
int stack::getTop()
{
    return top->getData();
}
bool stack::IsEmpty()
{
    if(top==NULL)
    {
        return true;
    }
    else
    return false;
}


int main()
{
    cout<<"Hello World! "<<endl;

cout<<"********************************\n"<<endl;
        cout<<"\tstack with array\n"<<endl;
cout<<"********************************\n"<<endl;
    Stack s(5);
//*****check stack is full or not and Empty or not*****
    cout<<"Is Empty= "<<s.IsEmpty()<<endl;
    cout<<"Is FULL = "<<s.IsFull()<<endl;
    //*****Push data into stack*****
    for(int i=1;i<=5;i++)
    {
        int a;
        cout<<"Enter number: ";
        cin>>a;
        s.push(a);
    }
    s.push(20);
    //*****After Pushing*****
//*****check stack is full or not and Empty or not*****
    cout<<"Is Empty= "<<s.IsEmpty()<<endl;
    cout<<"Is FULL = "<<s.IsFull()<<endl;
    //*****Get the value At the top of stack(only get vale at the top of stack
not remove)*****
    cout<<"Top = "<<s.getTop()<<endl;
    //*****Pop data From stack*****
    for(int i=1;i<=5;i++)
    {
        cout<<"Pop = "<<s.pop()<<endl;
    }
    //*****After Poping*****
//*****check stack is full or not and Empty or not*****
//*****Answer will be empty(0) and full (0)*****
    cout<<"Is Empty= "<<s.IsEmpty()<<endl;
    cout<<"Is FULL = "<<s.IsFull()<<endl;

cout<<"********************************\n"<<endl;
        cout<<"\tstack with Linked list\n"<<endl;
cout<<"********************************\n"<<endl;
```

```
stack s1;
    cout<<"Is Empty = "<<s1.IsEmpty()<<endl;
    s1.push(1);
    cout<<"Top = "<<s1.getTop()<<endl;
    s1.push(2);
    cout<<"Top = "<<s1.getTop()<<endl;
    s1.push(3);
    cout<<"Is Empty = "<<s1.IsEmpty()<<endl;
    cout<<"Top = "<<s1.getTop()<<endl;
    cout<<"Pop = "<<s1.pop()<<endl;
    cout<<"Top = "<<s1.getTop()<<endl;
    cout<<"Pop = "<<s1.pop()<<endl;
    cout<<"Is Empty = "<<s1.IsEmpty()<<endl;
    cout<<"Top = "<<s1.getTop()<<endl;
    cout<<"Pop = "<<s1.pop()<<endl;
    cout<<"Is Empty = "<<s1.IsEmpty()<<endl;

}
```

HOMEWORK 5:

Question:

1. Design a C++ program to implement the evaluation of a postfix expression using a stack. Implement a stack using a linked list, and provide functionalities to push, pop, and check the top element of the stack. The program should read a postfix expression as input and use the stack to evaluate the result. Support the basic arithmetic operators (+, -, *, /) and handle operand and operator errors appropriately.

```
#include<iostream>
#include<math.h>
using namespace std;
class stack
{
    private:
        int *ptr;
        int current;
        int length;
    public:
        stack(int);
        void push(int);
        int pop();
        int top();
        bool IsFull();
        bool IsEmpty();
};
stack::stack(int l)
{
    length=l;
    ptr=new int[length];
    current=-1;
}
void stack::push(int d)
{
    if(current==length-1)
    {
        cout<<"Stack is full"<<endl;
    }
    else
    {
    ptr[++current]=d;
    }
}
int stack::pop()
{
    if(current==-1)
    {
        return -1;
    }
    else
    {
    return ptr[current--];
    }
}
int stack::top()
{
    if(current==-1)
    return -1;
    else
    return ptr[current];
}
bool stack::IsEmpty()
{
    if(current==-1)
    {
        return true;
    }
    else
    return false;
}
bool stack::IsFull()
{
    if(current==length-1)
    {
        return true;
    }
    else
    return false;
}

int main()
{
    int i = 0, op1,op2,x;
    char input[100];
    stack s(10);

    cout<<"Enter a valid Postfix Expression: "<<endl;
    cin>>input;

    while(input[i] != '\0')
    {
        if(input[i]>='0' && input[i]<='9')
        {
            x = input[i] - 48;
            s.push(x);
        }
        else if(input[i]=='+' || input[i]=='-' || input[i]=='*' || input[i]=='/'||input[i]=='^')
        {
            op2 = s.pop();
            op1 = s.pop();
            switch(input[i]){
                case '+':
                    s.push(op1+op2);
                    break;
                case '-':
                    s.push(op1-op2);
                    break;
                case '*':
                    s.push(op1*op2);
                    break;
                case '/':
                    s.push(op1/op2);
                    break;
                case '^':
                    s.push(pow(op1,op2));
                    break;
            }
        }
        i++;
    }
    cout<<"Answer = "<<s.pop();
    return 0;
}

```

2. Design a C++ program to implement the evaluation of an infix expression using a stack. Implement a stack using a linked list, and provide functionalities to push, pop, and check the top element of the stack. The program should read an infix expression as input, convert it to postfix notation, and then use the stack to evaluate the result. Support the basic arithmetic operators (+, -, *, /) and parentheses. Handle operand and operator errors appropriately

```
#include<iostream>
#include<string>
#include<math.h>
using namespace std;
template <class t>
class stack
{
        private:
                int top;
                t arr[10];
        public:
                stack();
                void push(t);
                t pop();
                t topElement();
                bool IsEmpty();
};
template <class t>
stack<t>::stack()
{
        top=-1;
}
```

```
template <class t>
void stack<t>::push(t val)
{
        arr[++top]=val;
}
template <class t>
t stack<t>::topElement()
{
        return arr[top];
}
template <class t>
t stack<t>::pop()
{
        return arr[top--];
}
template <class t>
bool stack<t>::IsEmpty()
{
    if(top==-1)
    {
        return true;
    }
    else
    return false;
}
int prec (char c){
    if (c == '^'){
        return 3;
    }
    else if (c == '*' || c == '/'){
        return 2;
    }
    else if (c == '+' || c == '-'){
        return 1;
    }
    else {
        return -1;
    }
}
int main()
{
    int i = 0;
    char input[100];
    string res;
    stack<char> s;
    cout<<"Enter an Infix Expression: "<<endl;
    cin>>input;

    while(input[i] != '\0')
    {
        if(input[i]>='a' && input[i]<='z' || input[i]>='A' && input[i]<='Z')
        {
            res += input[i];
        }
        else if(input[i]=='(')
        {
            s.push(input[i]);
        }
        else if(input[i]==')')
        {
            while(!s.IsEmpty() && s.topElement()!='(' )
            {
            res += s.topElement();
            s.pop();
            }
            if(!s.IsEmpty()){
                s.pop();
            }
        }
        else {
            while (!s.IsEmpty() && prec(s.topElement()) > prec(input[i]))
            {
                res += s.topElement();
                s.pop();
            }
            s.push(input[i]);
        }
        i++;
        }
            while(!s.IsEmpty()){
            res += s.topElement();
            s.pop();
        }
    cout<<"Postfix form is: ";
    cout<<res<<endl;
    return 0;
}
```

HOMEWORK 6:

Design a C++ program to implement a queue using an array. Include functionalities to enqueue, dequeue, and check the front element of the queue. Ensure that the queue can handle a specified maximum number of elements to prevent overflow. Implement error handling for queue underflow conditions. Design a C++ program to implement a queue using a linked list. Defne a Node structure for the linked list and include functionalities to enqueue, dequeue, and check the front element of the queue. Implement error handling for queue underflow conditions

```
#include<iostream>
using namespace std;
#include<iostream>
using namespace std;

/***********************************
        Queue with Array
***********************************/
template <class t>
class queue
{
    private:
        t *data;
        int size;
        int front;
        int rear;
        int ele;
    public:
        queue(int);
        void enqueue(t);
        t dequeue();
        t getFront();
        int Elements();
        bool Isempty();
        bool IsFull();
};
template <class t>
queue<t>::queue(int s)
{
    size=s;
    front=1;
    rear=0;
    data=new t[size];
}
template <class t>
void queue<t>::enqueue(t x)
{
    rear=(rear+1)%size;
    data[rear]=x;
    ele++;
}
template <class t>
t queue<t>::dequeue()
{
    t x=data[front];
    front=(front+1)%size;
    ele--;
    return x;
}
template <class t>
t queue<t>::getFront()
{
    return data[front];
}
template <class t>
int queue<t>::Elements()
{
    return ele;
}
template <class t>
bool queue<t>::Isempty()
{
    if(front==0)
    return true;
    else
    return false;
}
template <class t>
bool queue<t>::IsFull()
{
    if(rear==size)
    return true;
    else
    return false;
}
```

```
/**********************************
        Queue with Linked List
**********************************/
template <class t>
class node
{
    private:
        t data;
        node<t> *next;
    public:
        node(t);
        void setData(t);
        t getData();
        void showData();
        void setNext(node<t>*);
        node<t>* getNext();
};
template <class t>
node<t>::node(t d)
{
    setData(d);
    next=NULL;
}
template <class t>
void node<t>::setData(t d)
{
    data=d;
}
template <class t>
t node<t>::getData()
{
    return data;
}
template <class t>
void node<t>::showData()
{
    cout<<data<<"   ";
}
template <class t>
void node<t>::setNext(node<t> *ptr)
{
    next=ptr;
}
template <class t>
node<t>* node<t>::getNext()
```

```
{
    return next;
}
template<class T>
class Queue
{
    private:
        node<T> *front;
        node<T> *rare;
    public:
        Queue();
        int enqueuq(T);
        T dequeue();
        T frontE();
        bool IsEmpty();
};
template<class T>
Queue<T>::Queue()
{
    front=NULL;
    rare=NULL;
}
template<class T>
int Queue<T>::enqueuq(T d)
{
    node<T> *ptr=new node<T>(d);
    if(front==NULL)
    {
        front=ptr;
        rare=ptr;
    }
    else
    //ptr->setNext(NULL);
    rare->setNext(ptr);
    rare=ptr;
    return 1;
}
template<class T>
T Queue<T>::dequeue()
{
    T x=front->getData();
    node<T>* ptr=front;
    front=front->getNext();
    delete ptr;
    return x;
```

```cpp
}
template<class T>
bool Queue<T>::IsEmpty()
{
    return (front==NULL);

}
template<class T>
T Queue<T>::frontE()
{
    return front->getData();
}
int main()
{
    cout<<"Hello world!"<<endl;

cout<<"***********************************\n"<<endl;
    cout<<"\tQueue with Array \n"<<endl;
cout<<"***********************************\n"<<endl;
    queue<int> i(5);
    i.enqueue(1);
    i.enqueue(2);
    i.enqueue(3);
    i.enqueue(4);
    i.enqueue(5);
    cout<<"Front   Element = "<<i.getFront()<<endl;
    cout<<"Dequeue Element = "<<i.dequeue()<<endl;
    cout<<"Dequeue Element = "<<i.dequeue()<<endl;
    cout<<"Is  Empty Queue = "<<i.Isempty()<<endl;
    cout<<"Is  Full  Queue = "<<i.IsFull()<<endl;
    cout<<"No.  of Element = "<<i.Elements()<<endl;
    i.enqueue(10);
    i.enqueue(20);
    cout<<"No.  of Element = "<<i.Elements()<<endl;
    cout<<"Front   Element = "<<i.getFront()<<endl;
    cout<<"Dequeue Element = "<<i.dequeue()<<endl;
    cout<<"Dequeue Element = "<<i.dequeue()<<endl;
    cout<<"Dequeue Element = "<<i.dequeue()<<endl;
    cout<<"Dequeue Element = "<<i.dequeue()<<endl;
    cout<<"Dequeue Element = "<<i.dequeue()<<endl;
    cout<<"No.  of Element = "<<i.Elements()<<endl;
    cout<<"Front   Element = "<<i.getFront()<<endl;
    cout<<"Is  Empty Queue = "<<i.Isempty()<<endl;
    cout<<"Is  Full  Queue = "<<i.IsFull()<<endl;
```

```
cout<<"***********************************\n"<<endl;
    cout<<"\tQueue with Linked List\n"<<endl;
cout<<"***********************************\n"<<endl;

    Queue<int> q;
    cout<<"Empty = "<<q.IsEmpty()<<endl;
    cout<<"Enque data = "<<q.enqueuq(1)<<endl;
    cout<<"Empty = "<<q.IsEmpty();
    cout<<endl;
    cout<<"Front Element = "<<q.frontE()<<endl;
    cout<<"Enqueuue data = "<<q.enqueuq(2)<<endl;
    cout<<"Dequeue = "<<q.dequeue()<<endl;
    cout<<"Front Element = "<<q.frontE()<<endl;
    cout<<"Dequeue = "<<q.dequeue()<<endl;
    cout<<"Empty = "<<q.IsEmpty();
    cout<<endl;
    cout<<"Enque data = "<<q.enqueuq(10)<<endl;
    cout<<"Enque data = "<<q.enqueuq(100)<<endl;
    cout<<"Enque data = "<<q.enqueuq(1000)<<endl;
    cout<<"Enque data = "<<q.enqueuq(10000)<<endl;
    cout<<"Enque data = "<<q.enqueuq(100000)<<endl;
    cout<<"Enque data = "<<q.enqueuq(1000000)<<endl;
    cout<<"Front Element = "<<q.frontE()<<endl;
    cout<<"Dequeue = "<<q.dequeue()<<endl;
    cout<<"Front Element = "<<q.frontE()<<endl;
    cout<<"Dequeue = "<<q.dequeue()<<endl;
    cout<<"Front Element = "<<q.frontE()<<endl;
    cout<<"Dequeue = "<<q.dequeue()<<endl;
    cout<<"Front Element = "<<q.frontE()<<endl;
    cout<<"Dequeue = "<<q.dequeue()<<endl;
    cout<<"Front Element = "<<q.frontE()<<endl;
    cout<<"Dequeue = "<<q.dequeue()<<endl;
    cout<<"Front Element = "<<q.frontE()<<endl;
    cout<<"Dequeue = "<<q.dequeue()<<endl;
    cout<<"Empty = "<<q.IsEmpty();
    cout<<"\n\n*************************\n*************************\n\n"<<endl;
    Queue<string> q1;
    cout<<"Empty = "<<q1.IsEmpty()<<endl;
    cout<<"Enque data = "<<q1.enqueuq("BIT21218")<<endl;
    cout<<"Empty = "<<q1.IsEmpty();
    cout<<endl;
    cout<<"Front Element = "<<q1.frontE()<<endl;
    cout<<"Enqueuue data = "<<q1.enqueuq("FARHAN AHMAD")<<endl;
    cout<<"Dequeue = "<<q1.dequeue()<<endl;
    cout<<"Front Element = "<<q1.frontE()<<endl;
    cout<<"Dequeue = "<<q1.dequeue()<<endl;
    cout<<"Empty = "<<q1.IsEmpty();
    cout<<endl;
    cout<<"Enque data = "<<q1.enqueuq("BIT21219")<<endl;
    cout<<"Enque data = "<<q1.enqueuq("BIT21220")<<endl;
    cout<<"Enque data = "<<q1.enqueuq("BIT21221")<<endl;
    cout<<"Enque data = "<<q1.enqueuq("BIT21222")<<endl;
    cout<<"Enque data = "<<q1.enqueuq("BIT21223")<<endl;
    cout<<"Enque data = "<<q1.enqueuq("BIT21224")<<endl;
    cout<<"Front Element = "<<q1.frontE()<<endl;
    cout<<"Dequeue = "<<q1.dequeue()<<endl;
    cout<<"Front Element = "<<q1.frontE()<<endl;
    cout<<"Dequeue = "<<q1.dequeue()<<endl;
    cout<<"Front Element = "<<q1.frontE()<<endl;
    cout<<"Dequeue = "<<q1.dequeue()<<endl;
    cout<<"Front Element = "<<q1.frontE()<<endl;
    cout<<"Dequeue = "<<q1.dequeue()<<endl;
    cout<<"Front Element = "<<q1.frontE()<<endl;
    cout<<"Dequeue = "<<q1.dequeue()<<endl;
    cout<<"Front Element = "<<q1.frontE()<<endl;
    cout<<"Dequeue = "<<q1.dequeue()<<endl;
    cout<<"Empty = "<<q1.IsEmpty();

}
```

HOMEWORK 7:

Question:

Design a C++ program to implement a binary tree. Defne a Node structure that includes data, a pointer to the left child, and a pointer to the right child. Implement functions to perform the operations.

CODE:

```
#include<iostream>
using namespace std;
class node
{
    private:
        int data;
        node *left;
        node *right;
    public:
        node(int);
        void setData(int);
        int getData();
        void showData();
        void setLeft(node*);
        node* getLeft();
        void setRight(node*);
        node* getRight();
        bool IsLeaf();
};
node::node(int d)
{
    setData(d);
    left=NULL;
    right=NULL;
}
void node::setData(int d)
{
    data=d;
}
int node::getData()
{
    return data;
}
void node::showData()
{
    cout<<data<<"  "<<endl;
}
void node::setLeft(node *ptr)
{
    left=ptr;
}
node* node::getLeft()
{
    return left;
}
void node::setRight(node *ptr)
{
    right=ptr;
}
node* node::getRight()
{
    return right;
```

```
}
bool node::IsLeaf()
{
    if(left==NULL&&right==NULL)
    return true;
    else
    return false;
}

void insert(node *root,int val)
{
    node *ptr=new node(val);
    node *p;
    node *q;
    p=root;
    q=root;
    while(q!=NULL&&p->getData()!=val)
    {
        p=q;
        if(val<p->getData())
        q=p->getLeft();
        else
        q=p->getRight();
    }
    if(p->getData()==val)
    cout<<"Attempt to insert duplicate value "<<val<<endl;
    else if(val<p->getData())
    {
        p->setLeft(ptr);
        cout<<val<<endl;
    }
    else
    {
        p->setRight(ptr);
        cout<<val<<endl;
    }
}
int main()
{
    cout<<"Hello World!"<<endl;
    int arr[]={14,15,4,9,7,18,3,5,16,4,20,17,9,14,5,-1};
    node *root=new node(arr[0]);
    for(int i=1;arr[i]>0;i++)
    {
        insert(root,arr[i]);
    }
}
```

Demonstrate the usage of your program by inserting nodes into the tree and then calling these recursive functions.

CODE:

```
#include<iostream>
#include<conio.h>
using namespace std;
class node
{
    private:
        int data;
        node *left;
        node *right;
    public:
        node(int );
        void set_data(int );
        int get_data();
        void set_left(node *);
        node* get_left();
        void set_right( node *);
        node* get_right();

};
node::node(int d)
{
    data = d;
    left = NULL;
    right = NULL;
}
void node::set_data(int d)
{
    data = d;
}
int node::get_data()
{
    return data;
}
void node::set_left(node *ptr)
{
    left = ptr;
}
node * node::get_left()
{
    return left;
}
void node::set_right(node * ptr)
{
    right = ptr;
}
node * node::get_right()
{
    return right;
}
////////////////////////////////////////////////////////////
//defintion of insert function
void insert(node *r,int val)
{
    node *newnode=new node(0);
    newnode->set_data(val);
    node *p,*f;
    p=f=r;
    while(p->get_data()!=val&&f!=NULL)
    {
        p=f;
        if(f->get_data()>val)
        f=f->get_right();
        else
        f=f->get_left();
    }
    if(val==p->get_data())
    {
    cout<<"\nSAME Value....."<<val<<endl;
    delete newnode;
    return;
    }
    else if(p->get_data()>val)
    p->set_right(newnode);
    else
    p->set_left(newnode);
}
          //definition the the preorder function
void preorder(node *r)
{
    if(r!=NULL)
    {
        cout<<"  "<<r->get_data();
        preorder(r->get_left()) ;
        preorder(r->get_right());
    }
}
          //definiton of the member function
void inorder(node *r)
{
    if(r!=NULL)
    {
        inorder(r->get_left())  ;
        cout<<r->get_data()<<"  ";
        inorder(r->get_right());
    }
}
          //definiton of the member function post order
void postorder(node *r)
{
    if(r!=NULL)
    {
        postorder(r->get_left())    ;
        postorder(r->get_right());
        cout<<r->get_data()<<"  ";
    }
}
//////////////////////////////////////////////////////////
//****Main bodey**************
int main()
{
    int a[7]={14,4,16,3,9,15,18};
    node *root=new node(0);
    root->set_data(a[0]);
    for(int i=1;i<7;i++)
    {
        insert(root,a[i])   ;
    }
          cout<<"\nDisplay data in preorder format...\n";
          preorder(root);
          cout<<"\nDisplay data in inorder format....\n";
          inorder(root);
          cout<<"\nDisplay data in postorder format....\n";
          postorder(root);
    getch();
   }
```

Demonstrate the usage of your program by inserting nodes into the tree and then calling these non-recursive functions.

CODE:

```cpp
#include<iostream>
using namespace std;
template<class T>
class node
{
    public:
        node(T d);
        void setData(T d);
        T getData();
        void setNext(node<T> * n);
        node<T> * getNext();
    private:
        T data;
        node<T> *next;
};
template <class T>
node<T>::node(T d)
{
    data = d;
    next = NULL;
}
template <class T>
void node<T>::setData(T d)
{
    data = d;
}
template <class T>
T node<T>::getData()
{
    return data;
}
template <class T>
void node<T>::setNext(node<T> *n)
```

```
{
    next = n;
}
template <class T>
node<T> * node<T>::getNext()
{
    return next;
}//End of Class Node

    //class stack
template <class T>
class stack
{
    private:
        node<T> * head;
    public:
        stack();
        void push(T);
        T pop();
        T top();
        int is_Empty();
        void show();
};
template <class T>
stack<T>::stack()
{
    head = NULL;
}
template <class T>
void stack<T>::push(T val)
{
    node<T> * ptr = new node<T>(val);
    ptr ->setNext(head);
    head = ptr;
}
template <class T>
T stack<T>::pop()
{
    node<T> *ptr = head;
    T val = head ->getData();
    head = head->getNext();
    delete ptr;
    return val;
}
template <class T>
```

```
T stack<T>::top()
{
    return head ->getData();
}
template <class T>
int stack<T>::is_Empty()
{
    if(head==NULL)
        return true;
    else
        return false;
}
template <class T>
void stack <T>::show()
{
    while(head!=NULL)
    {
        cout<<pop();
        cout<<endl;
    }
}
    //End of stack
template <class T>
class TreeNode
{
    private:
        T data;
        TreeNode<T> * left;
        TreeNode<T> * right;
    public:
        TreeNode(T);
        void setData(T);
        T getData();
        void setleft(TreeNode<T> *);
        TreeNode<T> * getLeft();
        void setRight(TreeNode<T> *);
        TreeNode<T> * getRight();
        int isLeaf();
};
template <class T>
TreeNode<T>::TreeNode(T d)
    {
        data=d;
        left=NULL;
        right=NULL;
    }
template <class T>
void TreeNode<T>::setData(T d)

    {
        data=d;
    }
template <class T>
T TreeNode<T>::getData()
    {
        return data;
    }
template <class T>
void TreeNode<T>::setleft(TreeNode<T> *l)
    {
        left=l;
    }
template <class T>
TreeNode<T> * TreeNode<T>::getLeft()
    {
        return left;
    }
template <class T>
void TreeNode<T>::setRight(TreeNode<T> *r)
    {
        right=r;
    }
template <class T>
TreeNode<T> * TreeNode<T>::getRight()
    {
        return right;
    }
template <class T>
int TreeNode<T>::isLeaf()
    {
        if(this->left==NULL && this->right==NULL)
            return 1;
        return 0;
    }// End of TreeNode Class
/////////////////////////////////////////////////////////////////////////////////
////////////////////////////////

                                //Global fn of insert
template <class T>
void insert(TreeNode<T> * root, T d)
```

```
    {
        TreeNode<T> * node=new TreeNode<T>(d);
        TreeNode<T> * p, * q;
        p=q=root;
        while(d!=p->getData() && q!= NULL)
            {
                p=q;
                if(d<p->getData())
                    q=p->getLeft();
                else
                    q=p->getRight();
            }
        if(d==p->getData())
            {
                cout<<"\nAttempt to Insert Duplicate : "<<d<<endl;
                delete node;
            }
        else if(d<p->getData())
            p->setleft(node);
        else
            p->setRight(node);
}//End Of Insert fn

                            //Global fn of preorder
template <class T>
void preorder(TreeNode<T> * node)
{
    stack <TreeNode<T> *> s;
    TreeNode<T> *p;
    p=node;
    do
    {
        while(p!=NULL)
        {
            cout<<p->getData()<<" ";
            s.push(p);
            p=p->getLeft();
        }
        if(!s.is_Empty())
        {
            p=s.pop();
            p=p->getRight();
        }
    }while(p!=NULL || !(s.is_Empty()));
}
```

```
                            //Global fn of inorder
template <class T>
void inorder(TreeNode<T> * node)
{
    stack<TreeNode<T>*> s;
    TreeNode<T> * p;
    p=node;
    do
    {
        while(p!=NULL)
        {
            s.push(p);
            p=p->getLeft();
        }
    if(!s.is_Empty())
    {
        p=s.pop();
        cout<<p->getData()<<" ";
        p=p->getRight();
    }
    }while(p!=NULL || !(s.is_Empty()));
}
        //Global fn of postorder
template <class T>
void postorder(TreeNode<T> * node)
{
    stack<TreeNode<T>*> s;
    stack<TreeNode<T>*> output;
    s.push(node);
    while (!s.is_Empty())
    {
        TreeNode<T> * curr = s.top();
        output.push(curr);
        s.pop();
        if (curr->getLeft())
        s.push(curr->getLeft());
        if (curr->getRight())
        s.push(curr->getRight());
    }
    while (!output.is_Empty())
    {
        cout << output.top()->getData() << " ";
        output.pop();
    }
}
int main()
{
    int x[]={14,4,3,9,16,15,18,-1};
    TreeNode<int> * root=new TreeNode<int>(14);
    for(int i=1;x[i]!=-1;i++)
    {
        insert(root,x[i]);
    }
    cout<<"\nPreorder  : ";
    preorder(root);
    cout<<"\nInorder   : ";
    inorder(root);
    cout<<"\nPostorder : ";
    postorder(root);
}
```

LABS

LAB 1:

Question:

1. **Suppose, we have a Shape class having following function.**
   virtual void display area() = 0;

2. **Shape class is inherited by three different classes, i.e. Square, Rectangle, and Triangle.**

3. **Each child class must implement the area function. (You may use your own logic to implement it).**

Formula of triangle area = (length * base) / 2
Formula of rectangle area = length * width
Formula of square area = length *length

4. **Now, you write the following main function and verify the output.**

Main function:

```
int main()
{
	Shape * s = new Triangle(5, 4);
	s->displayArea();
	delete s;
	s = new Rectangle(5, 4);
	s->displayArea();
	delete s;	s =
new Square(5);
	s-
>displayArea();
	delete s;
	return 0;
}
```

Output:

```
Area of triangle is 10
Area of rectangle is 20
Area of square is 25
```

CODE:

```
#include <iostream>
using namespace std;
class shape
```

```
{
public:
    virtual void displayarea() = 0;
};

class triangle : public shape
{
private:
    int length;
    int base;

public:
    triangle(int, int);
    void setlength(int);
    void setbase(int);
    void displayarea();
    int getlength();
    int getbase();
};

triangle::triangle(int len, int bas)
{
    setlength(len);
    setbase(bas);
}

void triangle::setlength(int len)
{
    length = len;
}

int triangle::getlength()
{
    return length;
}

void triangle::setbase(int bas)
{
    base = bas;
}

int triangle::getbase()
{
    return base;
}
```

```cpp
void triangle::displayarea()
{
    int area = (length * base) / 2;
    std::cout << "Area of triangle is " << area << std::endl;
}

class square : public shape {
private:
    int length1;
    int length2;

public:
    square(int, int);
    void setlength1(int);
    void setlength2(int);
    int getlength1();
    int getlength2();
    void displayarea();
};

square::square(int l1, int l2) {
    setlength1(l1);
    setlength2(l2);
}

void square::setlength1(int l1) {
    length1 = l1;
}

int square::getlength1() {
    return length1;
}

void square::setlength2(int l2) {
    length2 = l2;
}

int square::getlength2() {
    return length2;
}

void square::displayarea() {
    int area = length1 * length2;
    std::cout << "Area of square is " << area << std::endl;
}


class rectangle : public shape
{
private:
    int length;
    int width;

public:
    rectangle(int, int);
    void setlength(int);
    void setwidth(int);
    void displayarea();
    int getlength();
    int getwidth();
};

rectangle::rectangle(int len, int wid)
{
    setlength(len);
    setwidth(wid);
}

void rectangle::setlength(int len)
{
    length = len;
}

int rectangle::getlength()
{
    return length;
}

void rectangle::setwidth(int wid)
{
    width = wid;
}

int rectangle::getwidth()
{
    return width;
}

void rectangle::displayarea()
```

```cpp
{
    int area = length * width;
    std::cout << "Area of rectangle is " << area << std::endl;
}

int main()
{
    shape *s = new triangle(5, 5);
    s->displayarea();
    delete s;

    s = new rectangle(5, 6);
    s->displayarea();
    delete s;

    s = new square(5, 7);
    s->displayarea();
    delete s;

    return 0;
}
```

LAB 2:

Question:

In last lecture, we have learnt the Stack data structure with push and pop functions. Today, we will implement it in the c++ language.

Suppose, we have the following Stack class.

```
class Stack
{
private:
    int size;
int noOfElements;
int data[size];

public:
    Stack() {…} //constructor to initialize all the required data members
    Stack(int size) {…} //constructor to initialize all the required data members
bool push (int element) {…} //add the element in the stack
    int top () {…} //return the top of the stack without removing it from stack
int pop () {…} //return the top of the stack and also remove it from stack.
If stack is empty return -1
    int totalElements () {…} //return the total number of elements from the stack
void display() {…} //display all the elements of the stack from bottom to top };
```

1. You have to implement all its functions.

2. Now, you write the following main function and verify the output.

Main function:	Output
int main()	1
{	1
Stack s(4);	1
cout << s.push(5) << endl;	1
cout << s.push(10) << endl; cout	0
<< s.push(15) << endl; cout <<	20
s.push(20) << endl; cout <<	15
s.push(25) << endl; cout <<	1
s.pop() << endl; cout << s.pop()	30
<< endl; cout << s.push(30) <<	3
endl; cout << s.top() << endl;	Stack: 5, 10, 30
cout << s.totalElements() << endl;	
s.display(); return 0;	
}	

CODE:

```
#include <iostream>
using namespace std;

class Stack {
private:
    int size;
    int noOfElements;
    int *data;
```

```
public:
 Stack(int);
 ~Stack();
 bool push (int ); // add element to stack.
 int pop ();     // remove the top most element from stack and return true if
successful otherwise false..
 int top();
 int totalElements();
 void display();

 };
Stack::Stack(int stackSize) {
        size = stackSize;
        noOfElements = 0;
        data = new int[size];
    }

    Stack::~Stack() {
        delete[] data;
    }

    bool Stack:: push(int element) {
        if (noOfElements < size) {
            data[noOfElements] = element;
            noOfElements++;
            return true;
        } else {
            return false; // Stack is full
        }
    }

    int Stack:: top() {
        if (noOfElements > 0) {
            return data[noOfElements - 1];
        } else {
            return -1; // Stack is empty
        }
    }

    int Stack:: pop() {
        if (noOfElements > 0) {
            int poppedElement = data[noOfElements - 1];
            noOfElements--;
            return poppedElement;
        } else {
```

```cpp
            return -1; // Stack is empty
        }
    }

    int Stack:: totalElements() {
        return noOfElements;
    }

    void Stack:: display() {
        cout<<"stack:"<<"  ";
        for (int i = 0; i < noOfElements; i++) {
            cout << data[i] << "  , ";
        }
        cout << endl;
    }


int main() {
    Stack s(4);
    cout << s.push(5) << endl;    // 1 (true)
    cout << s.push(10) << endl;   // 1 (true)
    cout << s.push(15) << endl;   // 1 (true)
    cout << s.push(20) << endl;   // 1 (true)
    cout << s.push(25) << endl;   // 0 (false), stack is full
    cout << s.pop() << endl;      // 20
    cout << s.pop() << endl;      // 15
    cout << s.push(30) << endl;   // 1 (true)
    cout << s.top() << endl;      // 30
    cout << s.totalElements() << endl; // 3
    s.display(); // Displays: 5 10 30

    return 0;
}
```

LAB 3:

Question:

In last lecture, we have learnt the Stack data structure by dynamically allocate the memory. Today, we will implement it in the c++ language.

Suppose, we have the following Stack class.

```
class Stack
{
private:
   int size;     int
noOfElements;
int * data;

public:
   Stack() {…} //constructor to initialize all the required data members
   Stack(int size) {…} //constructor to initialize all the required data members
bool push (int element) {…} //add the element in the stack and if stack is full,
then add 5 new locations to it
   int top () {…} //return the top of the stack without removing it from stack
int pop () {…} //return the top of the stack and also remove it from stack.
If stack is empty return -1
   int totalElements () {…} //return the total number of elements from the stack
void display() {…} //display all the elements of the stack from bottom to top
   ~Stack() {…} //delete the allocated memory from
heap };
```

1. You have to implement all its functions.

2. Now, you write the following main function and verify the output.

Main function:	**Output**

int main()
{
   Stack * s = new Stack(3);   cout
<< s->push(5) << endl;   cout <<
s->push(10) << endl;   cout << s-
>push(15) << endl;   cout << s-
>push(20) << endl;   cout << s-
>push(25) << endl;   cout << s-
>pop() << endl;   cout << s->pop()
<< endl;   cout << s->push(30) <<
endl;   cout << s->top() << endl;
cout << s->totalElements() << endl;
s->display();   return 0;
}
	1
	1
	1
	1
	1
	25
	20
	1
	30
	4
	Stack: 5, 10, 15, 30

CODE:

```
#include <iostream>
using namespace std;
class Stack
{
private:
    int size;
    int noOfElements;
    int *data;

public:
    Stack(int);
    ~Stack(); // delete the allocated memory from heap
    bool push(int);

    int top(); // return the top of the stack without removing it from stack
    int pop(); // return the top of the stack and also remove it from stack.

    int totalElements(); // return the total number of elements from the stack
    void display();      // display all the elements of the stack from bottom to
top
};
Stack::Stack(int stacksize)
{
    size = stacksize;
    noOfElements = 0;     // initialize the count variable as zero initially;
    data = new int[size]; // dynamically allocate space for array in heap ;
}

    bool Stack:: push(int element) {
        if (noOfElements >= size) {
            // If stack is full, allocate new memory with increased size
            int newSize = size + 5;
            int *newData = new int[newSize];

            // Copy the elements from the old array to the new array
            for (int i = 0; i < size; i++) {
                newData[i] = data[i];
            }

            // Deallocate the old memory
            delete[] data;

            // Update the data pointer and size
            data = newData;
            size = newSize;
        }

        data[noOfElements] = element;
        noOfElements++;
        return true;
    }
int Stack::top()
{
    if (noOfElements > 0)
    {
        return data[noOfElements -1];
    }
    else
    {
        return -1;
    }
}
int Stack::pop()
{
    if (noOfElements > 0)
    {
        int poped = data[noOfElements - 1];
        noOfElements--;
        return poped;
    }
    else
    {
        return -1;
    }
}
int Stack ::totalElements()
{
    return noOfElements;
}
void Stack::display()
{
    cout << "stack:  ";
    for (int i = 0; i < noOfElements; i++)
    {

        cout << data[i] << "   ";
    }
}
```

```
int main()
{

    Stack *s = new Stack(3);
    cout << s->push(5) << endl;
    cout << s->push(10) << endl;
    cout << s->push(15) << endl;
    cout << s->push(20) << endl;
    cout << s->push(25) << endl;
    cout << s->pop() << endl;
    cout << s->pop() << endl;
    cout << s->push(30) << endl;
    cout << s->top() << endl;
    cout << s->totalElements() << endl;
    s->display();
    return 0;
}
```

LAB 4:

Question:

Today, you will implement the Link List class as we have discussed it in the previous lectures.

Suppose, we have the following Node and LinkList class.

```
class Node {
private:
    int value;
Node * next;

public:
    Node(int value){
this->value = value;
next = null;
    }
    void setValue (int value) {    this->value = value;    }
void setNext (Node * next) {    this->next = next;    }
    int getValue () {    return value;    }
Node* getNext () {    return next;    } };
```

```
class LinkList
{
private:
    Node * head;

public:
    LinkList () {…} //constructor to initialize the head node

    void insertAtTail (int element) {…} //add the element at the end of the link list

    int deleteFromTail () {…} //return and delete the last element from the link list
                               //if link list is empty, then return -1
     void insertAtHead (int element) {…} //add the element at the start of the link
list

    int deleteFromHead () {…} //return and delete the first element from the link list
                               //if link list is empty, then return -1

    void display () {…} //display the contents of the link list from head to tail

    int totalElements () {…} //return the total number of elements from the link list
    ~LinkList() {…} //delete the allocated memory from heap
};
```

1. You have to implement all functions of the both classes.
2. Now, you write the following main function and verify the output.

Main function:

```
int main()
{
    LinkList list;

list.insertAtTail(40);
list.insertAtTail(50);
list.insertAtTail(60);

list.insertAtHead(10);
list.insertAtHead(20);
list.insertAtHead(30);

    cout << "Total Elements: " << list.totalElements << endl;

    list.display();
     cout << list.deleteFromTail()
<<endl;     cout <<
list.deleteFromHead() <<endl;

    cout << "Total Elements: " << list.totalElements << endl;

    list.display();
}
```

Output :
```
Total Elements: 6
List: 30, 20, 10, 40, 50, 60
60
30
Total Elements: 4
List: 20, 10, 40, 50
```

CODE:

```cpp
#include <iostream>
using namespace std;

class node
{
private:
    int value;
    node *next;

public:
    node(int);
    void setvalue(int);
    int getvalue();
    void setnext(node *);
    node *getnext();
    void showdata();
};

node::node(int val)
{
    setvalue(val);
    next = NULL;
}

void node::setvalue(int val)
{
    value = val;
}

int node::getvalue()
{
    return value;
}

void node::setnext(node *ptr)
{
    next = ptr;
}

node *node::getnext()
{
    return next;
}

void node::showdata()
{
    cout << value << "    ";
}

class list
{
private:
    node *head;

public:
    list();
    void insertAtTail(int);
    int deleteFromTail();
    void insertAtHead(int);
    int deleteFromHead();
    void display();
    int totalElements();
};

list::list()
{
    head = NULL;
}

void list::insertAtTail(int elem)
{
    node *newNode = new node(elem);
    if (!head)
    {
        head = newNode;
    }
    else
    {
        node *current = head;
        while (current->getnext() != NULL)
        {
            current = current->getnext();
        }
        current->setnext(newNode);
    }
}

void list::insertAtHead(int elem)
{
    node *ptr = new node(elem);
    ptr->setnext(head);
    head = ptr;
}

int list::deleteFromHead()
{
    if (!head)
    {
        return -1;
    }
    else
    {
        int value = head->getvalue();
        node *ptr = head;
        head = head->getnext();
        delete ptr;
        return value;
    }
}

int list::deleteFromTail()
{
    if (!head)
    {
        return -1;
    }
    else if (head->getnext() == NULL)
    {
        int value = head->getvalue();
        delete head;
        head = NULL;
        return value;
    }
    else
    {
        node *current = head;
        while (current->getnext()->getnext() != NULL)
        {
            current = current->getnext();
        }
        int value = current->getnext()->getvalue();
        delete current->getnext();
        current->setnext(NULL);
        return value;
    }
}

int list::totalElements()
{
    node *ptr = head;
    int count = 0;
    while (ptr != NULL)
    {
        count++;
        ptr = ptr->getnext();
    }
    return count;
}

void list::display()
{
    node *current = head;
    while (current != NULL)
    {
        current->showdata();
        current = current->getnext();
    }
}

int main()
{
    list l;
    l.insertAtTail(40);
    l.insertAtTail(50);
    l.insertAtTail(60);
    l.insertAtHead(10);
    l.insertAtHead(20);
    l.insertAtHead(30);

    cout << "Total Elements: " << l.totalElements() << endl;
    l.display();
    cout << l.deleteFromTail() << endl;
    cout << l.deleteFromHead() << endl;
    cout << "Total Elements: " << l.totalElements() << endl;
    l.display();

    return 0;
}
```

LAB 5:

Today, you will implement the Circular Single Link List class as we have discussed it in the previous lectures.

Suppose, we have the following Node and CircularSingleLinkList class.

```
class Node { private:
   int value;     Node
* next;
 public:
   Node(int value){        this-
>value = value;        next = null;
   }
   void setValue (int value) {    this->value = value;    }     void
setNext (Node * next) {    this->next = next;    }
   int getValue () {    return value;    }     Node*
getNext () {    return next;    } };
```

```
class CircularSingleLinkList
{ private:
   Node * head;
 public:
   CircularSingleLinkList () {…} //constructor to initialize the head node

   void insertAtTail (int element) {…} //add the element at the end of the link list

   int deleteFromTail () {…} //return and delete the last element from the link list
                         //if link list is empty, then return -1
    void insertAtHead (int element) {…} //add the element at the start of the link list

   int deleteFromHead () {…} //return and delete the first element from the link list
                         //if link list is empty, then return -1

   void display () {…} //display the contents of the link list from head to tail

   int totalElements () {…} //return the total number of elements from the link list
   ~CircularSingleLinkList() {…} //delete the allocated memory from heap };
```

1. You have to implement all functions of the both classes.

2. Now, you write the following main function and verify the output.

```
Main function:
int main()
{
   CircularSingleLinkList list;
    list.insertAtTail(40);
list.insertAtTail(50);
list.insertAtTail(60);
    list.insertAtHead(10);
list.insertAtHead(20);
list.insertAtHead(30);

   cout << "Total Elements: " << list.totalElements << endl;

   list.display();
    cout << list.deleteFromTail() <<endl;
cout << list.deleteFromHead() <<endl;

   cout << "Total Elements: " << list.totalElements << endl;

   list.display();
}
```

```
Output :
Total Elements: 6
List: 30, 20, 10, 40, 50, 60
60
30
Total Elements: 4
List: 20, 10, 40, 50
```

CODE:

```
#include <iostream>
using namespace std;

class node
{
private:
    int value;
    node *next;

public:
    node(int);
    void setvalue(int);
    int getvalue();
    void setnext(node *);
    node *getnext();
    void showdata();
};

node::node(int val)
{
    setvalue(val);
    next = NULL;
}

void node::setvalue(int val)
{
    value = val;
}

int node::getvalue()
{
    return value;
}
```

```
void node::setnext(node *ptr)
{
    next = ptr;
}

node *node::getnext()
{
    return next;
}

void node::showdata()
{
    cout << value << "    ";
}

class list
{
private:
    node *head;

public:
    list();
    void insertathead(int);
    int delfromhead();
    void insertattail(int);
    int delfromtail();
    int totalelem();
    void display();
};

list::list()
{
    head = NULL;
}

void list::insertathead(int d)
{
    node *ptr = new node(d);
    if (!head)
    {
        head = ptr;
        ptr->setnext(head);
    }
    else
    {
        node *current = head;
        while (current->getnext() != head)
        {
            current = current->getnext();
        }

        ptr->setnext(head);
        current->setnext(ptr);
        head = ptr;
    }
}

int list::delfromhead()
{
    if (!head)
    {
        return -1;
    }
    else
    {
        int value = head->getvalue();
        if (head->getnext() == head)
        {
            delete head;
            head = NULL;
        }
        else
        {
            node *current = head;
            while (current->getnext() != head)
            {
                current = current->getnext();
            }
            node *temp = head;
            head = head->getnext();
            current->setnext(head);
            delete temp;
        }
        return value;
    }
}
void list::insertattail(int d)
{
    node *ptr = new node(d);
    if (!head)
    {
        head = ptr;
        ptr->setnext(head);
    }
    else
    {
        node *current = head;
        while (current->getnext() != head)
        {
            current = current->getnext();
        }
        current->setnext(ptr);
        ptr->setnext(head);
    }
}
int list::delfromtail()
{
    if (!head)
    {
        cout << "list is empty";
    }
    else
    {
        if (head->getnext() == head)
        {
            delete head;
            head = NULL;
        }
        else
        {
            node *current;
            node *temp;
            while (temp->getnext() != head)
            {
                temp = temp->getnext();
            };
            int value = temp->getvalue();
            while (current->getnext() != temp)
            {
                current = current->getnext();
            }
            delete temp;
            current->setnext(head);
            return value;
        }
    }
}
int list::totalelem()
{
    int count = 1;
    node *current = head;
    while (current->getnext() != head)
    {
        count++;
        current = current->getnext();
    }
    return count;
}
void list::display()
{
    if (!head)
    {
        cout << "Circular Linked List is empty." << endl;
        return;
    }

    node *current = head;
    cout << "list : ";
    do
    {
        cout <<current->getvalue() << " ";
        current = current->getnext();
    } while (current != head);
    cout << endl;
}

int main()
{
    list l;
 l.insertattail(40);
 l.insertattail(50);
 l.insertattail(60);
 l.insertathead(10);
 l.insertathead(20);
 l.insertathead(30);
 l.display();
  cout << l.delfromtail() <<endl;
 cout << l.delfromhead() <<endl;
 cout << "Total Elements: " << l.totalelem() << endl;
 l.display();

    return 0;
}
```

ASSIGNMENTS

ASSIGNMENT 1:

Question:

A case where circularly linked list comes in handy is the solution of the Josephus Problem. Consider there are 55 persons. They would like to choose two leaders “CR and GR” .

The way they decide is that all 55 sit in a circle. They start a count with person 1 and go in clockwise direction and skip 3. Person 4 reached is eliminated.

The count starts with the fifth and the next person to go is the fourth in count. Eventually, two persons remain

CODE:

```
#include<iostream>
#include<string.h>
using namespace std;
int limit[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
char
*m[13]={"null","January","February","March","April","May","June","July","August",
"September","October","November", "December"};
class date
{
    private:
            int day;
            int month;
            int year;
    public:
            date();
            date(int,int,int);
            void setDate(int,int,int);
            void setDay(int);
            void setMonth(int);
            void setYear(int);
            int getDay();
            int getMonth();
            int getYear();
            void printDate();
};
date::date()
{
}
date::date(int d,int m,int y)
```

```
{

        setMonth(m);
        setYear(y);
        setDay(d);
}
void date::setDay(int d)
{
        day = ((d>=1&&d<=limit[this->getMonth()])?d:1);
        if(this->getYear()%4==0&&this->getMonth()==2)
            day=((d>=1&&d<=29)?d:1);
}
void date::setMonth(int m)
{
        month =((m>=1&&m<=12)?m:1);
}
void date::setYear(int y)
{
        year = ((y>=1900&&y<=2023)?y:1900);
}
int date::getDay()
{
        return day;
}
int date::getYear()
{
        return year;
}
int date::getMonth()
{
        return month;
}
void date::printDate()
{
        cout<<this->getDay()<<":"<<m[this->getMonth()]<<":"<<this-
>getYear()<<endl;
}

class student
{
        private:
            char * id;
            char *name;
            date dob;
            char g;
        public:
            student();
            student(char *, char *, date, char);
            char getGender();
            void showStudent();

};
student::student()
{
}
student::student(char *i, char *n, date d, char gn)
{
        id = i;
        name = n;
        dob = d;
        g = gn;
}
void student::showStudent()
{
        cout<<"ID : "<< id<<endl;
        cout<<"Name : "<< name<<endl;
        cout<<"Date of birth : ";
        dob.printDate();
        cout<<"Gender : "<<g<<endl;

}
char student::getGender()
{
        return g;
}
class node
{
        private:
            student data;
            node * next;
        public:
            node(student );
            void setNext(node *);
            node* getNext();
            void showNode();
            student getData();
};

node::node(student d)
{
        data = d;
        next = NULL;
}
void node::setNext(node * ptr)
{
        next = ptr;
}
node* node::getNext()
{
        return next;
}
void node::showNode()
{
        data.showStudent();
}
student node::getData()
{
        return data;
}
class list
{
        private:
                node * head;
                node * current;
                int size;
                int maleCount;
                int femaleCount;
        public:
                list();
                void add(student d);
                void remove();
                void start();
                int getSize();
                void next();
                void showCurrent();
                void showList();
                int getFemaleCount();
                int getMaleCount();
                node* nodeToRemove();
};
list::list()
{
        head = NULL;
        current = NULL;
        size = 0;
```

```
}
void list::add(student d)
{
        node *ptr = new node(d);
        if(size == 0)
        {
                head = ptr;
                current = ptr;
                head->setNext(head);
        }
        else
        {
                ptr->setNext(current->getNext());
                current->setNext(ptr);
                current = ptr;
        }
        if(d.getGender()=='M')
        maleCount++;
        if(d.getGender()=='F')
        femaleCount++;
        size++;
}
void list::showList()
{
    node *ptr = head;
    do
    {
        ptr->showNode();
        ptr=ptr->getNext();
    }while(ptr!=head);
}
void list::remove()
{

    node *ptr = current->getNext();
    current->setNext(ptr->getNext());
//  current=ptr->getNext();
    if(ptr==head)
        head=head->getNext();

    student s = ptr->getData();
    if(s.getGender()=='M')
    maleCount--;
    if(s.getGender()=='F')
    femaleCount--;
    delete ptr;
    size--;
}
void list::start()
{
        current = head;
}
int list::getSize()
{
        return size;
}
void list::next()
{
        current=current->getNext();
}
void list::showCurrent()
{
        current->getNext()->showNode();
}
int list::getFemaleCount()
{
        return femaleCount;
}
int list::getMaleCount()
{
        return maleCount;
}
node* list::nodeToRemove()
{
        return current->getNext();
}
int main()
{
        date d1(15,4,1983);

        //s1.showStudent();
        date d2(15,5,1983);

        //s2.showStudent();
        student s1("MCSF02E001","Ali", d2,'M');
        student s2("MCSF02E002","Ahmad", d2,'M');
        student s3("MCSF02E003","Hasan", d2,'M');
        student s4("MCSF02E004","AR", d2,'M');
        student s5("MCSF02E005","IRHA", d2,'F');
        student s6("MCSF02E006","Eshaal", d2,'F');
        student s7("MCSF02E007","Iqra", d2,'F');
        student s8("MCSF02E008","Uzair", d2,'M');
        student s9("MCSF02E009","Salman",d1, 'M');
        student s10("MCSF02E010","Mahnoor Bloach", d2,'F');
        student s11("MCSF02E0011","Ijaz",d1, 'M');
        student s12("MCSF02E0012","Iqbal",d1, 'M');
        student s13("MCSF02E0013","Sharif",d1, 'M');
        student s14("MCSF02E0014","Romaisa",d1, 'F');
        student s15("MCSF02E0015","Aleeha",d1, 'F');
        student s16("MCSF02E0016","Arfa",d1, 'F');
        student s17("MCSF02E017","Khansa",d1, 'F');
        student s18("MCSF02E0018","Hamid",d1, 'M');
        student s19("MCSF02E0019","Meerab",d1, 'F');
        student s20("MCSF02E0020","Ibrahim",d1, 'M');
        student s21("MCSF02E0021","Adil",d1, 'M');
        student s22("MCSF02E0022","Adnan",d1, 'M');
        student s23("MCSF02E0023","Rabea",d1, 'F');
        student s24("MCSF02E0024","Asher",d1, 'M');
        student s25("MCSF02E0025","Arif",d1, 'M');

        list l;
        l.add(s1);
        l.add(s2);
        l.add(s3);
        l.add(s4);
        l.add(s5);
        l.add(s6);
        l.add(s7);
        l.add(s8);
        l.add(s9);
        l.add(s10);
        l.add(s11);
        l.add(s12);
        l.add(s13);
        l.add(s14);
        l.add(s15);
        l.add(s16);
        l.add(s17);
        l.add(s18);
        l.add(s19);
        l.add(s20);
        l.add(s21);
        l.add(s22);
        l.add(s23);
```

```
        l.add(s24);
        l.add(s25);
        l.start();
        l.showList();
        int i;
        node *ptr;
        student s;
        while(l.getSize()>2)
        {
                for(i=1;i<=3;i++)
                {
                        l.next();
                }

                ptr = l.nodeToRemove();
                s=ptr->getData();
                if(l.getFemaleCount()==1&&s.getGender()=='F')
                {
                    l.next();
                    continue;
                }
                if(l.getMaleCount()==1&&s.getGender()=='M')
                {
                    l.next();
                    continue;
                }

                cout<<"student to b removed : --------------"<<endl;
                l.showCurrent();
                l.remove();
        }
        cout<<endl<<endl<<"CR and GR data :-------------------- "<<endl;
        l.showList();

}
```

ASSIGNMENT 2:

Question:

Design a C++ program to calculate the age of a person given their birthdate. Follow these steps:

Allow the user to input their birthdate (day, month, and year).

Implement a function that calculates the age based on the current date.

Display the calculated age.

Ensure that your program handles diferent date formats, accounts for leap years, and validates input for reasonable birthdates.

Demonstrate the usage of your program by allowing the user to input their birthdate and then calculating and displaying their age.

CODE:

```
#include<iostream>
#include<string.h>
using namespace std;
int limit[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
string
name[13]={"Null","January","February","March","April","May","June","July","August
","September","October","November","December"};
class date
{
    private:
        int day;
        int month;
        int year;
    public:
        date(int ,int ,int );
        void setDate(int ,int ,int);
        void setDay(int);
        void setMonth(int);
        void setYear(int);
        int getDay();
        int getMonth();
        int getYear();
        void printDate();
        void printAge();
        void operator -=(date i)
        {
            if(day>i.day)
            day=day-i.day;
            else
            day=i.day-day;
            if(month>i.month)
            month=month-i.month;
            else
            month=i.month-month;
            year=i.year-year;
        }
};
date::date(int d,int m,int y)
{
    this->setDate(d,m,y);
}
void date::setDate(int d,int m,int y)
{
    this->setDay(d);
    this->setMonth(m);
    this->setYear(y);
}
void date::setDay(int d){
    day=((d>=1&&d<=limit[this->getMonth()])?d:1);
    if(this->getMonth()==2&&this->getYear()%4==0);
    day=((d>=1&&d<=29)?d:1);
}
void date::setMonth(int m){
        month=((m>=1&&m<=12)?m:1);
}
void date::setYear(int y){
        year=((y>=1990&&y<=2023)?y:1);
}
int date::getDay()
{
    return day;
}
int date::getMonth()
{
    return month;
}
int date::getYear()
{
    return year;
}

void date::printDate()
{
    cout<<"Date-of-Birth : "<<this->getDay()<<"-"<<name[this->month]<<"-"<<this-
>getYear()<<endl;
}
void date::printAge()
```

```
{
    cout<<"Current   Age : "<<this->getYear()<<"-"<<this->month<<"-"<<this-
>getDay()<<endl;
}

int main()
{
    int a,b,c,d,e,f;
    cout<<"Enter Date of Birth:"<<endl;
    cin>>a>>b>>c;
    cout<<"Enter Current Date :"<<endl;
    cin>>d>>e>>f;
    date dob(a,b,c);
    date cd(d,e,f);
    cout<<endl;
    dob.printDate();
    dob-=cd;
    dob.printAge();
}
```

ASSIGNMENT 3:

Question:

Design a C++ program to make the result card of student having class date for calculation of age, class data for course code, course name, marks and GPA, then make a class list for record of all students and then make class student for result card

CODE:

```
#include<iostream>
#include<string.h>
#include<iomanip>
using namespace std;
int limit[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
string
name[13]={"Null","January","February","March","April","May","June","July","August
","September","October","November","December"};
class date
{
    private:
        int day;
        int month;
        int year;
    public:
        date(int ,int ,int );
        void setDate(int ,int ,int);
        void setDay(int);
        void setMonth(int);
        void setYear(int);
        int getDay();
        int getMonth();
        int getYear();
        void printDate();
        void printAge();
        void operator -=(date i)
        {
            if(day>i.day)
            day=day-i.day;
            else
            day=i.day-day;
            if(month>i.month)
            month=month-i.month;
            else
            month=i.month-month;
            year=i.year-year;
        }
};
date::date(int d,int m,int y)
{
    this->setDate(d,m,y);
}
void date::setDate(int d,int m,int y)
{
    this->setDay(d);
    this->setMonth(m);
    this->setYear(y);
}
void date::setDay(int d){
    day=((d>=1&&d<=limit[this->getMonth()])?d:1);
    if(this->getMonth()==2&&this->getYear()%4==0);
    day=((d>=1&&d<=29)?d:1);
}
void date::setMonth(int m){
        month=((m>=1&&m<=12)?m:1);
}
```

```
void date::setYear(int y){
        year=((y>=1990&&y<=2023)?y:1);
}
int date::getDay()
{
    return day;
}
int date::getMonth()
{
    return month;
}
int date::getYear()
{
    return year;
}

void date::printDate()
{
    cout<<"Date-of-Birth : "<<this->getDay()<<"-"<<name[this->month]<<"-"<<this-
>getYear()<<endl;
}
void date::printAge()
{
    cout<<"Current   Age : "<<this->getYear()<<"-"<<this->month<<"-"<<this-
>getDay()<<endl;
}
/***********************************************
                class cource data
***********************************************/
class data
{
    private:
        string cCode;
        string cName;
        int marks;
        int crdhr;
        float gpa;
    public:
        data();
        data(string,string,int,int);
        setData(string,string,int,int);
        void setCode(string);
        void setName(string);
        void setMarks(int);
        void setCrdHr(int);
        string getCode();
        string getName();
        int getMarks();
        int getCrdHr();
        float getGradepoint();
        float getGPA();
        void printinformation();
};
data::data()
{
        cCode="00";
        cName="00";
        marks=0;
        crdhr=0;
        gpa=0.0;
}
data::data(string c,string n,int m,int cr)
{
        cCode=c;
        cName=n;
        marks=m;
        crdhr=cr;
        gpa=0.0;
}
data::setData(string c,string n,int m,int cr)
{
        setCode(c);
        setName(n);
        setMarks(m);
        setCrdHr(m);
}
void data::setCode(string c)
{
    cCode=c;
}
void data::setName(string n)
{
    cName=n;
}
void data::setMarks(int m)
{
   marks=m;
}
float data::getGradepoint()
{
    if(marks>=85&&marks<=100)
    return 4.00*crdhr;
    else if(marks>=80&&marks<=84)
    return 3.70*crdhr;
    else if(marks>=75&&marks<=79)
    return 3.30*crdhr;
    else if(marks>=70&&marks<=74)
    return 3.00*crdhr;
    else if(marks>=65&&marks<=69)
    return 2.70*crdhr;
    else if(marks>=61&&marks<=64)
    return 2.30*crdhr;
    else if(marks>=58&&marks<=60)
    return 2.00*crdhr;
    else if(marks>=55&&marks<=57)
    return 1.70*crdhr;
    else if(marks>=50&&marks<=54)
    return 1.00*crdhr;
    else
    return 0.00*crdhr;
}
void data::setCrdHr(int cr)
{
    crdhr=cr;
}
string data::getCode()
{
    return cCode;
}
string data::getName()
{
    return cName;
}
int data::getMarks()
{
    return marks;
}
int data::getCrdHr()
{
    return crdhr;
}
void data::printinformation()
{
    cout<<this->getCode()<<setw(40)<<setfill(' ')<<this-
>cName<<setw(10)<<setfill(' ')<<this->getMarks()<<setw(10)<<setfill(' ')<<this-
```

```
>getCrdHr()<<setw(10)<<setfill(' ')<<this->getGPA()<<setw(10)<<setfill('
')<<this->getGradepoint();
    cout<<endl;
}
float data::getGPA()
{
    if(marks>=85&&marks<=100)
    gpa= 4.00;
    else if(marks>=80&&marks<=84)
    gpa= 3.70;
    else if(marks>=75&&marks<=79)
    gpa= 3.30;
    else if(marks>=70&&marks<=74)
    gpa= 3.00;
    else if(marks>=65&&marks<=69)
    gpa= 2.70;
    else if(marks>=61&&marks<=64)
    gpa= 2.30;
    else if(marks>=58&&marks<=60)
    gpa= 2.00;
    else if(marks>=55&&marks<=57)
    gpa= 1.70;
    else if(marks>=50&&marks<=54)
    gpa= 1.00;
    else
    gpa= 0.00;
    return gpa;
}

/***********************************************
                class course list
***********************************************/

class list
{
    private:
        data *ptr;
        int current;
        int size;
        int length;
    public:
        list();
        list(int);
        void add(data);
        float getCGP();
        void printList();
        void copyList(list);
        void next();
        void back();
        void start();
        void tail();
        void clearList();
        int getLength();
        ~list();
};
list::list()
{
        length=0;
        size=0;
        current=0;
}
list::list(int l)
{
    if(l>0)
    {
        length=l+1;
        size=0;
        current=0;
        ptr=new data[length];

    }
}
void list::add(data val)
{
    if(current==length)
    {
        cout<<"list is full"<<endl;
    }
    else
    {
        ptr[++current]=val;
        size++;
    }
}
float list::getCGP()
{
    float sum=0.0;
    int cr=0;
    for(int i=1;i<=size;i++)
    {
        sum=sum+ptr[i].getGradepoint();
        cr=cr+ptr[i].getCrdHr();
    }
       return sum/cr;
}
void list::printList()
{
    if(size==0)
    {
        cout<<"List is empty :"<<endl;
    }
    cout<<"*********************************************************************
********************\n";
    cout<<"Cource code \t\t Subject \t\t\t Marks \t Cr.hr \t G_P\t T_G_P"<<endl;
    cout<<"*********************************************************************
********************\n";

    for(int i=1;i<=size;i++)
    {
    ptr[i].printinformation();
    }
    cout<<endl;
    cout<<"\tGPA = "<<this->getCGP()<<endl;
    }
void list::next()
{
    if(current==size)
    {
        cout<<"current is alread at end, cant move further"<<endl;
    }
    else
    current++;
}
void list::back()
{
    if(current==1)
    {
        cout<<"current is alread at start, cant move further"<<endl;
    }
    else
    current--;
}
void list::start()
{
    current==1;
```

```
}
void list::tail()
{
    current==size;
}
void list::clearList()
{
    current=0;
    size=0;
}
void list::copyList(list a)
{
    length=a.length;
    current=a.current;
    size=a.size;
    ptr=new data[length];
    for(int i=1;i<=length;i++)
        {
            ptr[i]=a.ptr[i];
        }
}
int list::getLength()
{
    return length;
}
list::~list()
{
    delete []ptr;
}

/**********************************************
                class Student
**********************************************/

class student
{
    private:
        string id;
        string name;
        string address;
        char gender;
        list *ptr;
        int current;
        int size;
        int length;
    public:
        student(int);
        void add(list);
        void setData(string,string,string,char);
        void showData();
        void setId(string);
        void setName(string);
        void setAddress(string);
        void setGender(char);
        string getId();
        string getName();
        string getAddress();
        char getGender();
};
student::student(int l)
{
    if(l>0)
    {
        length=l+1;
        size=0;
        current=0;
        ptr=new list[length];
    }
}
void student::setData(string Id,string Name,string Address,char Gender)
{
    id=Id;
    name=Name;
    address=Address;
    gender=Gender;
}
void student::setId(string Id)
{
    id=Id;
}
void student::setName(string Name)
{
    name=Name;
}
void student::setAddress(string Address)
{
    address=Address;
}
void student::setGender(char Gender)
{
    gender=Gender;
}
string student::getId()
{
    return id;
}
string student::getName()
{
    return name;
}
string student::getAddress()
{
    return address;
}
char student::getGender()
{
    return gender;
}
void student::add(list val)
{
    if(current==length)
    {
        cout<<"list is full"<<endl;
    }
    else
    {
        ptr[++current]=val;
        size++;
    }
}
void student::showData()
{
    cout<<"\tStudent ID : "<<id<<"\t\tName : "<<name<<"\t\tGender :
"<<gender<<endl;
    cout<<"\n\tAddress : "<<address<<"\t\tProgram : BSIT"<<endl;
    if(size==0)
    {
        cout<<"List is empty :"<<endl;
    }
    else
    {
    float sum;
    for(int i=1;i<=size;i++)
    {
    ptr[i].printList();
    cout<<"  ";
    sum=sum+ptr[i].getCGP();
    }
    cout<<endl;
    cout<<"CGPA= "<<sum/size;
    }
}
int main()
{
    int d,m,y,cd,cm,cy;
    char gender;
    string roll,name,address;
    cout<<"Enter Student ID: ";
    getline(cin,roll);
    cout<<"Enter Student Name: ";
    getline(cin,name);
    cout<<"Enter Student Address: ";
    getline(cin,address);
    cout<<"Enter Student Gender: ";
    cin>>gender;
    cout<<"Enter Date of Birth: ";
    cin>>d>>m>>y;
    cout<<"Enter Current Date: ";
    cin>>cd>>cm>>cy;

//******************Semester 1**************************
    data s1d1,s1d2,s1d3,s1d4,s1d5,s1d6;
    s1d1.setData("GE-161","Intro. to infor. & Comm. Technology",75,3);
    s1d2.setData("MS-152 ","Probability     and     Statistics",68,3);
    s1d3.setData("GE-162","English composition & Comprehension",84,3);
    s1d4.setData("CC-111 ","Discrete                 Structure",88,3);
    s1d5.setData("MS-151 ","Applied                    Physics",52,3);
    s1d6.setData("HQ-001 ","Quranic                Translation",82,0);

        list lS1(6);
        lS1.add(s1d1);
        lS1.add(s1d2);
        lS1.add(s1d3);
        lS1.add(s1d4);
        lS1.add(s1d5);
        lS1.add(s1d6);
        //*****************Semester 2**************************
    data s2d1,s2d2,s2d3,s2d4,s2d5,s2d6;
    s2d1.setData("CC-212  ","Programming           Fundamentls",88,4);
    s2d2.setData("CC-215  ","Database                    System",77,4);
    s2d3.setData("GE-165  ","Pakistan                  Studies",80,3);
    s2d4.setData("GE-164  ","Communication &Presentation skill",75,3);
    s2d5.setData("UE-171  ","Introduction     to     Economics",80,3);
    s2d6.setData("QT-002  ","Quran                 Translation",89,1);
    list lS2(6);
    lS2.add(s2d1);
    lS2.add(s2d2);
    lS2.add(s2d3);
    lS2.add(s2d4);
    lS2.add(s2d5);
    lS2.add(s2d6);
    //******************Semester 3**************************
    data s3d1,s3d2,s3d3,s3d4,s3d5,s3d6;
    s3d1.setData("CC-211 ","  Object   Oriented    Programming",75,4);
    s3d2.setData("UE-273 ","  Introduction    to     Sociology",68,3);
    s3d3.setData("MS-152 ","  Calculus and Analytical Geometry",84,3);
    s3d4.setData("EI-231 ","Computer Organization and Assembly",88,3);
    s3d5.setData("CC-214 ","  Computer                Networks",52,3);
    s3d6.setData("HQ-003 ","  Quran                Translation",80,2);
    list lS3(6);
    lS3.add(s3d1);
    lS3.add(s3d2);
    lS3.add(s3d3);
    lS3.add(s3d4);
    lS3.add(s3d5);
    lS3.add(s3d6);
    //**********************************************

    student s(3);
    s.add(lS1);
    s.add(lS2);
    s.add(lS3);
    s.setData(roll,name,address,gender);
    date i(d,m,y);
    date j(cd,cm,cy);
    cout<<"\n\n\t";
    i.printDate();
    i-=j;
    cout<<"\t\t";
    i.printAge();
    cout<<endl;
    s.showData();
}
```

ASSIGNMENT 4:

Question:

Design a C++ program to make the result card of student having class date for calculation of age, class data for course code, course name, marks and GPA, then make a class list for record of all students and then make class student for result card using linked list

CODE:

```
#include<iostream>
#include<iomanip>
#include<string.h>
using namespace std;
int limit[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
string
name[13]={"Null","January","February","March","April","May","June","July","August
","September","October","November","December"};

/************************************************
                    class date
************************************************/
class date
{
    private:
        int day;
        int month;
        int year;
    public:
        date(int ,int ,int );
        void setDate(int ,int ,int);
        void setDay(int);
        void setMonth(int);
        void setYear(int);
        int getDay();
        int getMonth();
        int getYear();
        void printDate();
        void printAge();
        void operator -=(date i)
        {
            if(day>i.day)
            day=day-i.day;
            else
            day=i.day-day;
            if(month>i.month)
            month=month-i.month;
            else
            month=i.month-month;
            year=i.year-year;
        }
};
date::date(int d,int m,int y)
{
    this->setDate(d,m,y);
}
void date::setDate(int d,int m,int y)
{
    this->setDay(d);
    this->setMonth(m);
    this->setYear(y);
}
void date::setDay(int d){
    day=((d>=1&&d<=limit[this->getMonth()])?d:1);
    if(this->getMonth()==2&&this->getYear()%4==0);
    day=((d>=1&&d<=29)?d:1);
}
void date::setMonth(int m){
        month=((m>=1&&m<=12)?m:1);
}
void date::setYear(int y){
        year=((y>=1990&&y<=2023)?y:1);
}
int date::getDay()
{
    return day;
}
int date::getMonth()
{
    return month;
}
int date::getYear()
{
    return year;
}

void date::printDate()
{
    cout<<"Date-of-Birth : "<<this->getDay()<<"-"<<name[this->month]<<"-"<<this-
>getYear();
}
void date::printAge()
{
    cout<<"Current   Age : "<<this->getYear()<<"-"<<this->month<<"-"<<this-
>getDay();
}
/************************************************
                    class data
************************************************/
class data
{
    private:
        string cCode;
        string cName;
        float marks;
        int crdhr;
        float gpa;
        data *next;
    public:
        data(string,string,float,int);
        void setCode(string);
        void setName(string);
        void setMarks(float);
        void setCrdHr(int);
        void setNext(data *);
        data *getNext();
        string getCode();
        string getName();
        float getMarks();
        int getCrdHr();
        float getGradepoint();
        float getGPA();
        void printinformation();
};
data::data(string c,string n,float m,int cr)
{
        cCode=c;
        cName=n;
        marks=m;
        setMarks(m);
        crdhr=cr;
        gpa=0.0;
}
```

```
void data::setCode(string c)
{
    cCode=c;
}
void data::setName(string n)
{
    cName=n;
}
void data::setMarks(float m)
{
   marks=m;
}
void data::setNext(data *ptr)
{
    next=ptr;
}
data* data::getNext()
{
    return next;
}
float data::getGradepoint()
{
    if(marks>=85&&marks<=100)
    return 4.00*crdhr;
    else if(marks>=80&&marks<=84)
    return 3.70*crdhr;
    else if(marks>=75&&marks<=79)
    return 3.30*crdhr;
    else if(marks>=70&&marks<=74)
    return 3.00*crdhr;
    else if(marks>=65&&marks<=69)
    return 2.70*crdhr;
    else if(marks>=61&&marks<=64)
    return 2.30*crdhr;
    else if(marks>=58&&marks<=60)
    return 2.00*crdhr;
    else if(marks>=55&&marks<=57)
    return 1.70*crdhr;
    else if(marks>=50&&marks<=54)
    return 1.00*crdhr;
    else
    return 0.00*crdhr;
}
void data::setCrdHr(int cr)
{
    crdhr=cr;
}
string data::getCode()
{
    return cCode;
}
string data::getName()
{
    return cName;
}
float data::getMarks()
{
    return marks;
}
int data::getCrdHr()
{
    return crdhr;
}
void data::printinformation()
{
cout<<this->getCode()<<setw(40)<<setfill(' ')<<this->cName<<setw(15)<<setfill('
')<<this->getMarks()<<setw(10)<<setfill(' ')<<this-
>getCrdHr()<<setw(10)<<setfill(' ')<<this->getGPA()<<setw(10)<<setfill('
')<<this->getGradepoint();
        cout<<endl;
}
float data::getGPA()
{
    if(marks>=85&&marks<=100)
    gpa= 4.00;
    else if(marks>=80&&marks<=84)
    gpa= 3.70;
    else if(marks>=75&&marks<=79)
    gpa= 3.30;
    else if(marks>=70&&marks<=74)
    gpa= 3.00;
    else if(marks>=65&&marks<=69)
    gpa= 2.70;
    else if(marks>=61&&marks<=64)
    gpa= 2.30;
    else if(marks>=58&&marks<=60)
    gpa= 2.00;
    else if(marks>=55&&marks<=57)
    gpa= 1.70;
    else if(marks>=50&&marks<=54)
    gpa= 1.00;
    else
    gpa= 0.00;
    return gpa;
}

/************************************************
                    class list
*************************************************/

class list
{
    private:
        data *head;
        data *current;
        int size;
        list *next;
    public:
        list();
        void add(string,string,float,int);
        void showList();
        float getCGP();
        void setNext(list *);
        list *getNext();
};
list::list()
{
    size=0;
    head=NULL;
    current=NULL;
}
void list::setNext(list *ptr)
{
    next=ptr;
}
list* list::getNext()
{
    return next;
}
void list::add(string c,string n,float m,int cr)
{
    data* ptr=new data(c,n,m,cr);
    if(size==0)
    {
        head=ptr;
        current=ptr;
    }
    else
    {
        ptr->setNext(current->getNext());
        current->setNext(ptr);
        current=ptr;
    }
    size++;
}
void list::showList()
{
    data* ptr=head;
    cout<<"**********************************************************************
***********************\n";
    cout<<"Cource code \t\t Subject \t\t\t Marks \t   Cr.hr\tG_P\tT_G_P"<<endl;
    cout<<"**********************************************************************
***********************\n";
    while(ptr!=NULL)
    {
        ptr->printinformation();
        ptr=ptr->getNext();
    }
    cout<<endl;
    cout<<"GPA = "<<this->getCGP()<<endl;
    cout<<endl;
}
float list::getCGP()
{
    data* ptr=head;
    float sum=0.0;
    int cr=0;
    while(ptr!=NULL)
    {
        sum=sum+ptr->getGradepoint();
        cr=cr+ptr->getCrdHr();
        ptr=ptr->getNext();
    }
    return sum/cr;
}

/***********************************************
                    class student
*************************************************/
class student
```

```
{
    private:
        string id;
        string name;
        string address;
        char gender;
        list *head;
        list *current;
        int Size;
    public:
        student();
        void add(list);
        void setData(string,string,string,char);
        void showStudent();
        void setId(string);
        void setName(string);
        void setAddress(string);
        void setGender(char);
        string getId();
        string getName();
        string getAddress();
        char getGender();
};
student::student()
{
    Size=0;
    head=NULL;
    current=NULL;
    id="NULL";
    name="NULL";
    address="NULL";
}
void student::setData(string Id,string Name,string Address,char Gender)
{
    id=Id;
    name=Name;
    address=Address;
    gender=Gender;
}
void student::setId(string Id)
{
    id=Id;
}
void student::setName(string Name)
{
    name=Name;
}
void student::setAddress(string Address)
{
    address=Address;
}
void student::setGender(char Gender)
{
    gender=Gender;
}
string student::getId()
{
    return id;
}
string student::getName()
{
    return name;
}
string student::getAddress()
{
    return address;
}
char student::getGender()
{
    return gender;
}
void student::add(list d)
{
    list* ptr=new list(d);
    if(Size==0)
    {
        head=ptr;
        current=ptr;
    }
    else
    {
        ptr->setNext(current->getNext());
        current->setNext(ptr);
        current=ptr;
    }
    Size++;
}
void student::showStudent()
{
    cout<<"\tStudent ID : "<<id<<"\t\\ttName : "<<name<<"\t\tGender :
"<<gender<<endl;
    cout<<"\n\tAddress : "<<address<<"\t\tProgram : BSIT"<<endl;
    if(Size==0)
    {
        cout<<"List is empty :"<<endl;
    }
    else
    {
    list* ptr=head;
    while(ptr!=NULL)
    {
        ptr->showList();
        ptr=ptr->getNext();
    }
    }
}
int main()
{
    string S1code[7]={"Null","GE-161","MS-152","GE-162","CC-111","MS-151","HQ-
001"};
    string S1name[7]={"Null","Intro.to infor. & Comm. Technology","Probability
and Statistics","English composition & Comprehension","Discrete
Structure","Applied Physics","Quranic Translation"};
    int S1chr[7]={0,3,3,3,3,3,0};
    string S2code[7]={"Null","CC-212","CC-215","GE-165","GE-164","UE-171","QT-
002"};
    string S2name[7]={"Null","Programming Fundamentls","Database
System","Pakistan Studies","Communication &Presentation skill","Introduction to
Economics","Quran Translation"};
    int S2chr[7]={0,4,4,3,3,3,1};
    string S3code[7]={"Null","CC-211","UE-273","MS-152","EI-231","CC-214","HQ-
003"};
    string S3name[7]={"Null","Object Oriented Programming","Introduction to
Sociology","Calculus and Analytical Geometry","Computer Organization and
Assembly","Computer Networks","Quran Translation"};
    int S3chr[7]={0,4,3,3,4,4,0};
    char gender;
    int d,m,y,cd,cm,cy;
    string roll,name,address;
    cout<<"Enter Student ID: ";
    getline(cin,roll);
    cout<<"Enter Student Name: ";
    getline(cin,name);
    cout<<"Enter Student Address: ";
    getline(cin,address);
    cout<<"Enter Student Gender: ";
    cin>>gender;
    cout<<"Enter Date of Birth: ";
    cin>>d>>m>>y;
    cout<<"Enter Current Date: ";
    cin>>cd>>cm>>cy;
    cout<<endl;
    cout<<"*****************Semester 1*************************"<<endl;
    cout<<"\n--------------Enter Data of 1st Semester--------------\n"<<endl;
    list lst1;
    for(int i=1;i<=6;i++){
        int marks;
        cout<<"Enter "<<S1name[i]<<"  Marks : ";
        cin>>marks;
        lst1.add(S1code[i],S1name[i],marks,S1chr[i]);
    }
    cout<<"\n*****************Semester 2*************************\n"<<endl;
    cout<<"\n--------------Enter Data of 2nd Semester--------------\n"<<endl;
    list lst2;
    for(int i=1;i<=6;i++){
        int marks;
        cout<<"Enter "<<S2name[i]<<"  Marks : ";
        cin>>marks;
        lst2.add(S2code[i],S2name[i],marks,S2chr[i]);
    }
    cout<<"\n*****************Semester 3*************************\n"<<endl;
    cout<<"\n--------------Enter Data of 3rd Semester--------------\n"<<endl;
    list lst3;
    for(int i=1;i<=6;i++){
        int marks;
        cout<<"Enter "<<S3name[i]<<"  Marks : ";
        cin>>marks;
        lst3.add(S3code[i],S3name[i],marks,S3chr[i]);
    }
    student s;
    s.add(lst1);
    s.add(lst2);
    s.add(lst3);
    s.setData(roll,name,address,gender);
    date i(d,m,y);
    date j(cd,cm,cy);
    cout<<"\n\n\t";
    i.printDate();
    i-=j;
    cout<<"\t\t";
    cout<<"Final CGPA = "<<(lst1.getCGP()+lst2.getCGP()+lst3.getCGP())/3;
    cout<<"   ";
    i.printAge();
    cout<<endl;
    s.showStudent();
}
```

ASSIGNMENT 5:

Question:

Design a C++ program to implement a doubly circular linked list. Defne a Node structure with data, a pointer to the next node, and a pointer to the previous node. Implement functions to perform the operations. Demonstrate the usage of your program by performing various insertions, deletions, traversals, and searches on the doubly circular linked list

CODE:

```
#include<iostream>
using namespace std;
class node
{
        private:
            int data;
            node *next;
            node *prev;
        public:
            node(int);
            void setData(int );
            int getData();
            void setNext(node *);
            node* getNext();
            void setPrev(node *);
            node* getPrev();
            void showData();
};
node::node(int d)
{
        setData(d);
        next=NULL;
        prev=NULL;
}
void node::setData(int d)
{
        data = d;
}
int node::getData()
{
        return data;
}
void node::setNext(node *ptr)
{
        next = ptr;
}
node* node::getNext()
{
        return next;
}
void node::setPrev(node *ptr)
{
        prev = ptr;
}
node* node::getPrev()
{
        return prev;
}
void node::showData()
{
        cout<<data<<"   ";
}
class list
{
        private:
            node *head;
            node *current;
            int size;
        public:
            list();
            void add(int );
            void showList();
            void remove();
            void insertAtHead(int);
};
list::list()
{
        head = NULL;
        current = NULL;
        size =0;
}
void list::add(int d)
{
        node *ptr = new node(d);
        if(size == 0){
                head = ptr;
                current = ptr;
                current->setNext(head);
                current->setPrev(head);
        }
        else{
                ptr->setNext(current->getNext());
                ptr->setPrev(current);
                current->getNext()->setPrev(ptr);
                current->setNext(ptr);
                current = ptr;
        }
        size++;
}
void list::insertAtHead(int d)
{
    node *ptr=new node(d);
    ptr->setNext(head);
    ptr->setPrev(current);
    current->setNext(ptr);
    current->getNext()->setPrev(ptr);
    head=ptr;
    current=ptr;
    size++;
}
void list::showList()
{
    if(size==0)
    {
        cout<<"List is Empty";
    }
        node *ptr=head;
        do
        {
            ptr->showData();
            ptr=ptr->getNext();
        }while(ptr!=head);
```

```
}
void list::remove()
{
    if(size==0)
    {
        cout<<"List is Empty";
    }
    if(head==current)
    {
        head=head->getNext();
    }
    node *temp=current;
    node *ptr=head;
    do{
        if(head==current)
        {
            break;
        }
        ptr=ptr->getNext();
    }
    while(ptr!=head);
    ptr->setNext(current->getNext());
    current=current->getNext();
    current->getNext()->setPrev(ptr);
    delete temp;
    size--;
}
int main()
{
        cout<<"hello world"<<endl;
        list l;
        l.add(40);
        l.showList();
        cout<<endl;
        l.add(90);
        l.showList();
        cout<<endl;
        l.add(91);
        l.showList();
        cout<<endl;
        l.add(30);
        l.showList();
        cout<<endl;
        l.insertAtHead(10);
        l.showList();
        cout<<endl;
        l.add(20);
        l.showList();
        l.remove();
        cout<<endl;
        l.showList();
        l.remove();
        cout<<endl;
        l.showList();
        l.remove();
        cout<<endl;
        l.showList();
        l.remove();
        cout<<endl;
        l.showList();
        l.remove();
        cout<<endl;
        l.showList();
        l.remove();
        cout<<endl;
l.add(40);
        l.showList();
        cout<<endl;
        l.add(90);
        l.showList();
        cout<<endl;
        l.add(91);
        l.showList();
        cout<<endl;
        l.add(30);
        l.showList();
        cout<<endl;
        l.insertAtHead(10);
        l.showList();
        cout<<endl;
        l.add(20);
        l.showList();
        l.remove();
        cout<<endl;
        l.showList();
        l.remove();
        cout<<endl;
        l.showList();
        l.remove();
        cout<<endl;
        l.showList();
        l.remove();
        cout<<endl;
        l.showList();
        l.remove();
        cout<<endl;
        l.showList();
        l.remove();
        cout<<endl;
}
```

ASSIGNMENT 6:

Question:

Design a C++ program to implement a generic queue using an array with templates. Include functionalities to enqueue, dequeue, and check the front element of the queue. Implement error handling for queue underflow conditions. In addition, use templates to make the queue suitable for storing elements of any data type. Demonstrate the usage of your generic queue implementation by enqueuing and dequeuing elements of various data types

CODE:

```
#include<iostream>
using namespace std;
template <class t>
class queue
{
    private:
        t *data;
        int size;
        int front;
        int rear;
        int ele;
    public:
        queue(int);
        int enqueue(t);
        t dequeue();
        t getFront();
        int Elements();
        bool Isempty();
        bool IsFull();
};
template <class t>
queue<t>::queue(int s)
{
    size=s;
    front=1;
    rear=0;
    data=new t[size];
}
template <class t>
int queue<t>::enqueue(t x)
{
    rear=(rear+1)%size;
    data[rear]=x;
    ele++;
    return 1;
}
template <class t>
t queue<t>::dequeue()
{
    t x=data[front];
    front=(front+1)%size;
    ele--;
    return x;
}
template <class t>
t queue<t>::getFront()
{
    return data[front];
}
template <class t>
int queue<t>::Elements()
{
    return ele;
}
template <class t>
bool queue<t>::Isempty()
{
    if(ele==0)
    return true;
    else
    return false;
}
template <class t>
```

```cpp
bool queue<t>::IsFull()
{
    if(ele==size)
    return true;
    else
    return false;
}
//*******************
int main()
{
    queue<int> i(5);
    cout<<"Enqueue data = "<<i.enqueue(1)<<endl;
    cout<<"Enqueue data = "<<i.enqueue(2)<<endl;
    cout<<"Enqueue data = "<<i.enqueue(3)<<endl;
    cout<<"Enqueue data = "<<i.enqueue(4)<<endl;
    cout<<"Enqueue data = "<<i.enqueue(5)<<endl;
    cout<<"Front   Element = "<<i.getFront()<<endl;
    cout<<"Dequeue Element = "<<i.dequeue()<<endl;
    cout<<"Dequeue Element = "<<i.dequeue()<<endl;
    cout<<"Is  Empty Queue = "<<i.Isempty()<<endl;
    cout<<"Is  Full  Queue = "<<i.IsFull()<<endl;
    cout<<"No.  of Element = "<<i.Elements()<<endl;
    cout<<"Enqueue data = "<<i.enqueue(10)<<endl;
    cout<<"Enqueue data = "<<i.enqueue(20)<<endl;
    cout<<"No.  of Element = "<<i.Elements()<<endl;
    cout<<"Front   Element = "<<i.getFront()<<endl;
    cout<<"Dequeue Element = "<<i.dequeue()<<endl;
    cout<<"Dequeue Element = "<<i.dequeue()<<endl;
    cout<<"Dequeue Element = "<<i.dequeue()<<endl;
    cout<<"Dequeue Element = "<<i.dequeue()<<endl;
    cout<<"Dequeue Element = "<<i.dequeue()<<endl;
    cout<<"No.  of Element = "<<i.Elements()<<endl;
    cout<<"Front   Element = "<<i.getFront()<<endl;
    cout<<"Is  Empty Queue = "<<i.Isempty()<<endl;
    cout<<"Is  Full  Queue = "<<i.IsFull()<<endl;
    cout<<"\n***********************\n***********************\n"<<endl;
    queue<string> i1(5);
    cout<<"Enqueue data = "<<i1.enqueue("BIT21218")<<endl;
    cout<<"Enqueue data = "<<i1.enqueue("BIT21218")<<endl;
    cout<<"Enqueue data = "<<i1.enqueue("BIT21218")<<endl;
    cout<<"Enqueue data = "<<i1.enqueue("BIT21218")<<endl;
    cout<<"Enqueue data = "<<i1.enqueue("BIT21218")<<endl;
    cout<<"Front   Element = "<<i1.getFront()<<endl;
    cout<<"Dequeue Element = "<<i1.dequeue()<<endl;
    cout<<"Dequeue Element = "<<i1.dequeue()<<endl;
    cout<<"Is  Empty Queue = "<<i1.Isempty()<<endl;
    cout<<"Is  Full  Queue = "<<i1.IsFull()<<endl;
    cout<<"No.  of Element = "<<i1.Elements()<<endl;
    cout<<"Enqueue data = "<<i1.enqueue("BIT21218")<<endl;
    cout<<"Enqueue data = "<<i1.enqueue("BIT21218")<<endl;
    cout<<"No.  of Element = "<<i1.Elements()<<endl;
    cout<<"Front   Element = "<<i1.getFront()<<endl;
    cout<<"Dequeue Element = "<<i1.dequeue()<<endl;
    cout<<"Dequeue Element = "<<i1.dequeue()<<endl;
    cout<<"Dequeue Element = "<<i1.dequeue()<<endl;
    cout<<"Dequeue Element = "<<i1.dequeue()<<endl;
    cout<<"Dequeue Element = "<<i1.dequeue()<<endl;
    cout<<"No.  of Element = "<<i1.Elements()<<endl;
    cout<<"Front   Element = "<<i1.getFront()<<endl;
    cout<<"Is  Empty Queue = "<<i1.Isempty()<<endl;
    cout<<"Is  Full  Queue = "<<i1.IsFull()<<endl;
}
```

ASSIGNMENT 7:

Question:

Design a C++ program to implement a generic queue using a linked list with templates. Define a Node structure for the linked list and include functionalities to enqueue, dequeue, and check the front element of the queue. Implement error handling for queue underflow conditions. Additionally, use templates to make the queue suitable for storing elements of any data type. Demonstrate the usage of your generic queue implementation by enqueuing and dequeuing elements of various data types

CODE:

```
#include<iostream>
using namespace std;
template <class t>
class node
{
    private:
        t data;
        node<t> *next;
    public:
        node(t);
        void setData(t);
        t getData();
        void showData();
        void setNext(node<t>*);
        node<t>* getNext();
};
template <class t>
node<t>::node(t d)
{
    setData(d);
    next=NULL;
}
template <class t>
void node<t>::setData(t d)
{
    data=d;
}
template <class t>
t node<t>::getData()
{
    return data;
}
template <class t>
void node<t>::showData()
{
    cout<<data<<"  ";
}
template <class t>
void node<t>::setNext(node<t> *ptr)
{
    next=ptr;
}
template <class t>
node<t>* node<t>::getNext()
{
    return next;
}
template<class T>
class queue
{
    private:
        node<T> *front;
        node<T> *rare;
    public:
        queue();
        int enqueuq(T);
        T dequeue();
        T frontE();
        bool IsEmpty();
};
template<class T>
queue<T>::queue()
{
    front=NULL;
    rare=NULL;
}
template<class T>
int queue<T>::enqueuq(T d)
{
    node<T> *ptr=new node<T>(d);
    if(front==NULL)
    {
        front=ptr;
        rare=ptr;
    }
    else
    //ptr->setNext(NULL);
    rare->setNext(ptr);
    rare=ptr;
    return 1;
}
template<class T>
T queue<T>::dequeue()
{
    T x=front->getData();
    node<T>* ptr=front;
    front=front->getNext();
    delete ptr;
    return x;
}
template<class T>
bool queue<T>::IsEmpty()
{
    return (front==NULL);

}
template<class T>
T queue<T>::frontE()
{
    return front->getData();
```

```cpp
}
int main()
{
    cout<<"Hello world!"<<endl;
    queue<int> q;
    cout<<"Empty = "<<q.IsEmpty()<<endl;
    cout<<"Enque data = "<<q.enqueuq(1)<<endl;
    cout<<"Empty = "<<q.IsEmpty();
    cout<<endl;
    cout<<"Front Element = "<<q.frontE()<<endl;
    cout<<"Enqueuue data = "<<q.enqueuq(2)<<endl;
    cout<<"Dequeue = "<<q.dequeue()<<endl;
    cout<<"Front Element = "<<q.frontE()<<endl;
    cout<<"Dequeue = "<<q.dequeue()<<endl;
    cout<<"Empty = "<<q.IsEmpty();
    cout<<endl;
    cout<<"Enque data = "<<q.enqueuq(10)<<endl;
    cout<<"Enque data = "<<q.enqueuq(100)<<endl;
    cout<<"Enque data = "<<q.enqueuq(1000)<<endl;
    cout<<"Enque data = "<<q.enqueuq(10000)<<endl;
    cout<<"Enque data = "<<q.enqueuq(100000)<<endl;
    cout<<"Enque data = "<<q.enqueuq(1000000)<<endl;
    cout<<"Front Element = "<<q.frontE()<<endl;
    cout<<"Dequeue = "<<q.dequeue()<<endl;
    cout<<"Front Element = "<<q.frontE()<<endl;
    cout<<"Dequeue = "<<q.dequeue()<<endl;
    cout<<"Front Element = "<<q.frontE()<<endl;
    cout<<"Dequeue = "<<q.dequeue()<<endl;
    cout<<"Front Element = "<<q.frontE()<<endl;
    cout<<"Dequeue = "<<q.dequeue()<<endl;
    cout<<"Front Element = "<<q.frontE()<<endl;
    cout<<"Dequeue = "<<q.dequeue()<<endl;
    cout<<"Front Element = "<<q.frontE()<<endl;
    cout<<"Dequeue = "<<q.dequeue()<<endl;
    cout<<"Empty = "<<q.IsEmpty();
    cout<<"\n\n*************************\n*************************\n\n"<<endl;
    queue<string> q1;
    cout<<"Empty = "<<q1.IsEmpty()<<endl;
    cout<<"Enque data = "<<q1.enqueuq("BIT21218")<<endl;
    cout<<"Empty = "<<q1.IsEmpty();
    cout<<endl;
    cout<<"Front Element = "<<q1.frontE()<<endl;
    cout<<"Enqueuue data = "<<q1.enqueuq("FARHAN AHMAD")<<endl;
    cout<<"Dequeue = "<<q1.dequeue()<<endl;
    cout<<"Front Element = "<<q1.frontE()<<endl;
    cout<<"Dequeue = "<<q1.dequeue()<<endl;
    cout<<"Empty = "<<q1.IsEmpty();
    cout<<endl;
    cout<<"Enque data = "<<q1.enqueuq("BIT21219")<<endl;
    cout<<"Enque data = "<<q1.enqueuq("BIT21220")<<endl;
    cout<<"Enque data = "<<q1.enqueuq("BIT21221")<<endl;
    cout<<"Enque data = "<<q1.enqueuq("BIT21222")<<endl;
    cout<<"Enque data = "<<q1.enqueuq("BIT21223")<<endl;
    cout<<"Enque data = "<<q1.enqueuq("BIT21224")<<endl;
    cout<<"Front Element = "<<q1.frontE()<<endl;
    cout<<"Dequeue = "<<q1.dequeue()<<endl;
    cout<<"Front Element = "<<q1.frontE()<<endl;
    cout<<"Dequeue = "<<q1.dequeue()<<endl;
    cout<<"Front Element = "<<q1.frontE()<<endl;
    cout<<"Dequeue = "<<q1.dequeue()<<endl;
    cout<<"Front Element = "<<q1.frontE()<<endl;
    cout<<"Dequeue = "<<q1.dequeue()<<endl;
    cout<<"Front Element = "<<q1.frontE()<<endl;
    cout<<"Dequeue = "<<q1.dequeue()<<endl;
    cout<<"Front Element = "<<q1.frontE()<<endl;
    cout<<"Dequeue = "<<q1.dequeue()<<endl;
    cout<<"Empty = "<<q1.IsEmpty();

}
```

ASSIGNMENT 8:

Question:

Design a C++ program to implement a tree structure using a dictionary. Use a dictionary data structure to represent a hierarchical structure where each node has a word and a meaning associated with it

CODE:

```
#include<iostream>
#include<conio.h>
#include<string.h>
#include<iomanip>
using namespace std;
class node
{
    private:
        string word;
        string meaning;
        node *left;
        node *right;
    public:
        node(string,string);
        void set_data(string,string);
        void show_data();
        void set_word(string);
        string get_word();
        void set_meaning(string);
        string get_meaning();
        void set_left(node *);
        node* get_left();
        void set_right(node*);
        node* get_right();
        bool isLeaf();
};
node::node(string d,string m)
{
    word = d;
    meaning = m;
    left = NULL;
    right = NULL;
}
void node::set_data(string d,string m)
{
    word = d;
    meaning = m;
}
void node::show_data()
{
    cout<<word<<"\t\t"<<meaning<<endl;
}
void node::set_word(string d)
{
    word = d;
}
string node::get_word()
{
    return word;
}
void node::set_meaning(string m)
{
    meaning = m;
```

```
}
string node::get_meaning()
{
    return meaning;
}
void node::set_left(node *ptr)
{
    left = ptr;
}
node* node::get_left()
{
    return left;
}
void node::set_right(node * ptr)
{
    right = ptr;
}
node * node::get_right()
{
    return right;
}
bool node::isLeaf()
{
        if(left==NULL && right==NULL)
        return true;
        else
        return false;
}
   //function to add new node in binary tree
void insert (node* r,string word,string meaning) ;
     //functions to display the binary tree values in
     //preorder....inorder.....postorder....
void preorder(node *);
void inorder(node *);
void postorder(node *);

//defintion of insert function
void insert(node *r,string word,string meaning)
{
    node *newnode=new node(word,meaning);
    node *p,*f;
    p=f=r;
    while(strcmp(word,p->get_word())!=0&& f !=NULL)
    {
        p=f;
        if(word>p->get_word())
        f=f->get_right();
        else
        f=f->get_left();
    }
    if(word==p->get_word())
    {
    cout<<"\nDuplicate Value....."<<word<<endl;
    delete newnode;
    return;
    }
    else if(word>p->get_word())
    p->set_right(newnode);
    else
    p->set_left(newnode);
}
//definition the the preorder function
void preorder(node *r)
{
    if(r!=NULL)
    {
    r->show_data();
    preorder(r->get_left()) ;
    preorder(r->get_right());
    }
}
//definiton of the member function
void inorder(node *r)
{
    if(r!=NULL)
    {
    inorder(r->get_left())  ;
    r->show_data();
    inorder(r->get_right());
    }
}
//definiton of the member function post order
void postorder(node *r)
{
    if(r!=NULL)
    {
    postorder(r->get_left())    ;
    postorder(r->get_right());
    r->show_data();
    }
```

```
}
            //****Main body**************
int main()
{
//      int a[7];
//      int b[7];
    string
a[100]={"about","abstract","almost","animated","backbone","backside","bad","becom
ing","candy","charter","cheesy","chorus"," dune "," dazzling "," domain ","
dignity "," excite "," eradicate "," evil "," ethical "," faith "," firm ","
forsake "," foreign "," galvanized ","
geared","garbage","gay","hall","handsome","happily","hard","idler","if","illibera
l","immediate","jealous","joy","jerkweed","knowingly","lacking","last","leading",
"legitimate","madness","magician","material","manmade","naked","nameless","napkin
","necessary","obdurate","object","obligatory","oblique","painting","particular",
"passable","pattern","quite","reasonable","refrain","reliable","religious","remai
nder","remark","reminiscence","sacrity","satisfied","scarcity","scrumptious","sec
ond","select","selection","signal","tailored","temper","terror","testament","ulti
mate","uncommon","uncooked","undeniable","vacancy","vague","vain","valueless","wa
rranty","well-timed","winery","well-
mannered","unhurt","unfortunate","unlawful","unmarried","trustworthy","twister","
twosome","zenith"};
    string
b[100]={"approximate","summary","nearly","lively","spine","behind","poor","fittin
g","sweet","constitution","corny","refrain"," mound "," blurring "," field ","
grace "," activate "," remove "," vice "," proper "," belief "," staunch ","
desert "," alien "," arouse "," ready ","rubbish","homosexual","corridor","good-
looking","fortunately","tough","loafer","whether","intolerant","instant","envious
","detestable","delight","deliberately","missing","final","main","valid","insanit
y","conjuror","fabric","artifical","bare","anonymous","serviette","essential","st
ubborn","thing","compulsory","indirect","portray","specific","satisfactory","samp
le","fair","fair","chorus","dependable","devout","rest","comment","memory","vestr
y","convinced","shortage","delicious","moment","choose","choice","sign","tailor-
made","mood","terrorism","testimony","final","unusual","raw","indisputable","empt
iness","indistinct","useless","worthless","guarantee","timelt","vineyard","polite
","unharmed","unlucky","illegal","single","reliable","tornado","pair","peak",};
    node *root=new node("NULL","NULL");
//      node<int> *root=new node<int>(0,0);
    root->set_data(a[0],b[0]);
//      cout<<"Word\t\tSynonym"<<endl;
    for(int i=1;i<100;i++)
    {
//          cin>>a[i];
//          cout<<"\t\t";
//          cin>>b[i];
```

```
        insert(root,a[i],b[i]);
    }
    cout<<"\nDisplay data in preorder format...\n";
    cout<<"Word\t\tSynonym\n"<<endl;
    preorder(root);
    cout<<"\nDisplay data in inorder format....\n";
    cout<<"Word\t\tSynonym\n"<<endl;
    inorder(root);
    cout<<"\nDisplay data in postorder format....\n";
    cout<<"Word\t\tSynonym\n"<<endl;
    postorder(root);
}
```

PROJECT REPORT HOTEL MANAGEMENT SYSTEM

PART 1: INTRODUCTION

1.1 Synopsis Project

Our C++ mini project is the Hotel Booking System. The project is for the admin of the hotel to manage the room and the booking details of customers and for the customer to book the hotel room. The type of data structures used in this system are sorting, linked list and queue. There are several classes in this system which are customer, bill and hotel room class. The admin can add, delete and check the availability of the hotel room with the linked list data structures which add or delete the room to the list. The room types have single, double, premium and deluxe. The customer can book, cancel the room and check the availability of the hotel room by updating the linked list of rooms. They have to fill in their personal details in order to book the room. The system updates the information of customers booking by linked list and sorts it by using the sorting technique. The customer booking is added to the queue of pending booking. After the customer booked the room, the admin can check the customer booking and review the booking information. The admin needs to confirm the pending booking and provide the bill for the customer by using queue data structure. The queue is for the pending booking to add to the bill queue for confirmation of booking and delete the last pending booking if want to cancel.

1.2 Objective of The Project

- For admin to manage the details of hotel room, booking and customers
- To simplify the booking process of customer

1.3 Languages used

C++ & Data structure and Algorithms (DSA)

1.4 Tools used

DEV C++ & VS CODE

PART 2: SYSTEM ANALYSIS AND DESIGN (USE CASE, FLOWCHART AND CLASS DIAGRAM)

2.1 System Requirements

2.1.1 Use case diagram

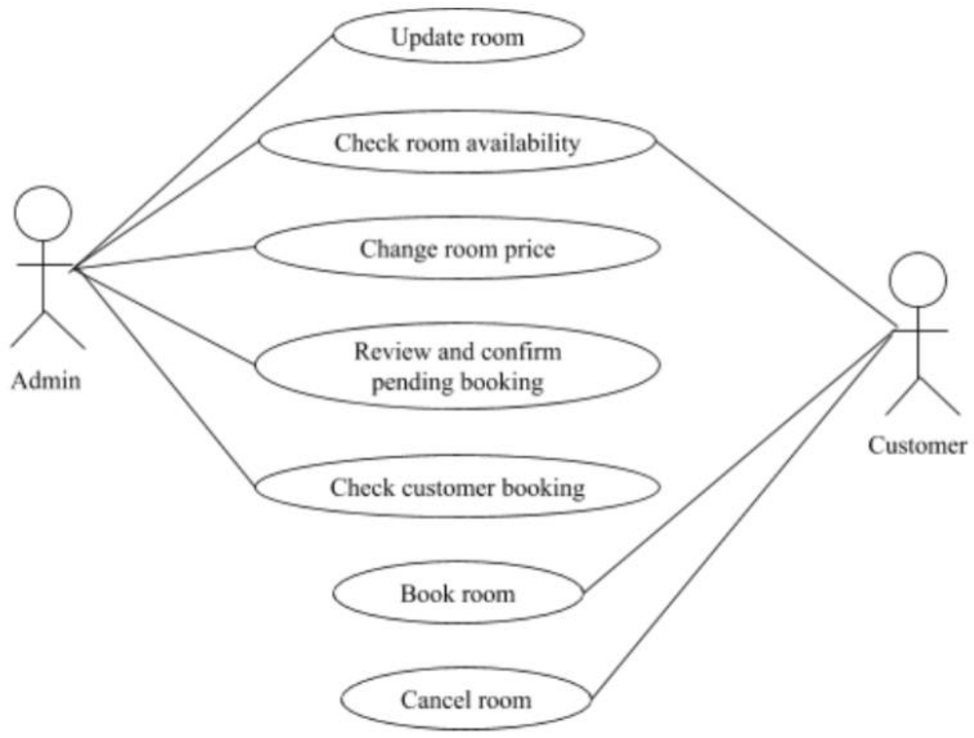


Figure 1: Use case diagram for Hotel Booking System

2.1.2 Use Cases Description for Hotel Booking System

The system users are admin and customer.

Actor	**Task**
Admin	The admin can add, delete and check the availability of the hotel room. The admin also can change the room price. After the customer booked the room, the admin can check the customer booking and review the booking information. The admin needs to confirm the pending booking and provide the bill for the customer.
Customer	The customer can book, cancel the room and check the availability of the hotel room. They have to fill in their personal details in order to book the room.

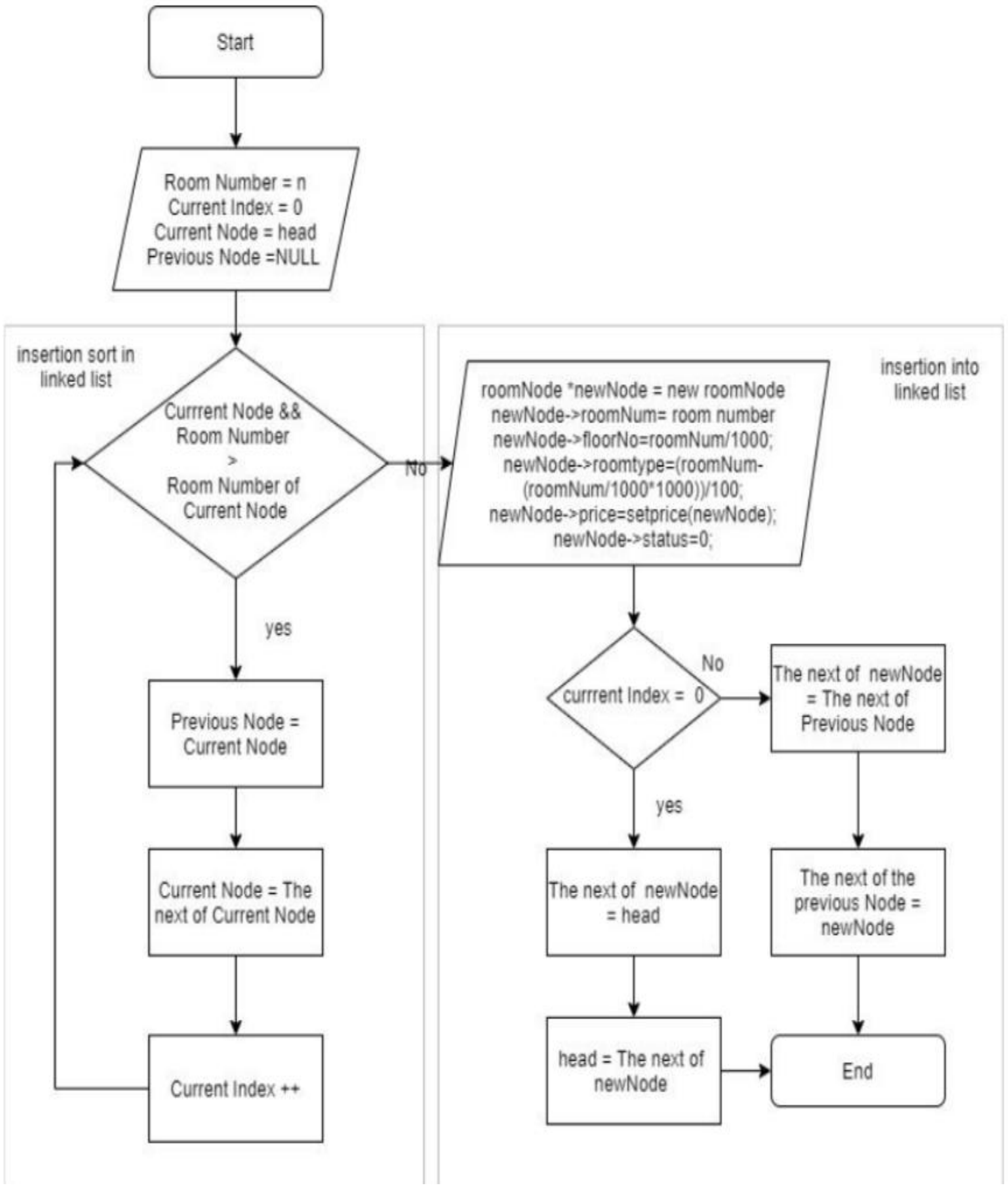
2.1.3 Detail Description for Each Use Cases

The system has 6 main use cases

Use Case	**Purpose**
Update room	Update information of room includes the updated room price by adding or deleting the room.
Check room availability	View the room information and availability of the room
Change room price	Change and update the price of each type of room
Check customer booking	View the detail list of the customer's booking
Review and confirm pending booking	Show all the details of the customers and the room before confirm the booking
Book room	Book the room and fill in their personal details by adding the booking to the booking list.
Cancel room	Make cancellation of their booking by deleting the booking from the booking list.

2.2 System Design

Flow Chart 1: Add room (Admin - Update room)

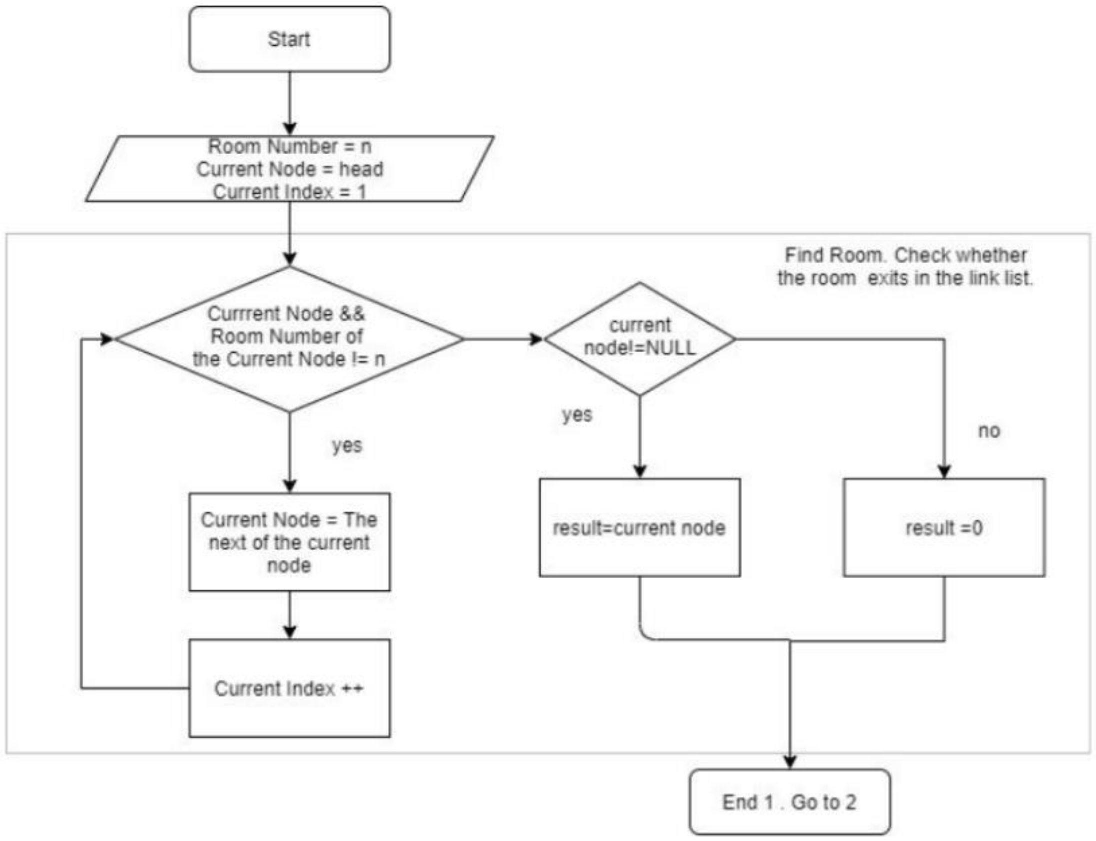

Start
Room Number = n
Current Index = 0
Current Node = head
Previous Node =NULL
insertion sort in linked list
Currrent Node && Room Number > Room Number of Current Node
No
yes
Previous Node = Current Node
Current Node = The next of Current Node
Current Index ++
insertion into linked list
roomNode *newNode = new roomNode
newNode->roomNum= room number
newNode->floorNo=roomNum/1000;
newNode->roomtype=(roomNum-(roomNum/1000*1000))/100;
newNode->price=setprice(newNode);
newNode->status=0;
currrent Index = 0
No
yes
The next of newNode = The next of Previous Node
The next of the previous Node = newNode
The next of newNode = head
head = The next of newNode
End

Explanation based on flowchart 1:

The data structure applied in this flowchart is insertion sort and linked list. The user is required to enter the room number that he/she wants to add to the system. It will first check whether the room number exists in the system or not. If the room number is unique, it will start the flow chart. The flowchart starts with initializing room number=n, current index = 0, points the current node to the head of the room linked list and the previous node set to NULL. The flowchart is divided into two-part, insertion sort and add a node into the room linked list. In the insertion sort part, it will find the correct position for the inserted node in ascending order of the room number. The flow chart starts to point current nodes to the head of the linked list. If the condition is true (the current node and the room number inserted by the user are greater than the room number of the current node), the loop will continue by changing the previousNode=currentNode, currentNode= the next of the current node and current index=current index+1. The loop will continue until the condition is false. When the condition is false, it will start the part to insert a node into the linked list. It starts with creating a new node. If the current index is equal to 0, it is an empty linked list, the next of the new node is pointing to the head (NULL) and the head is pointing to the new node. Otherwise, the next of the new node is pointing to the next of the previous node and the next of the previous node is pointing to the new node.

Flow Chart 2: Delete room (Admin - Update room)

Data Structure: Delete node in linked

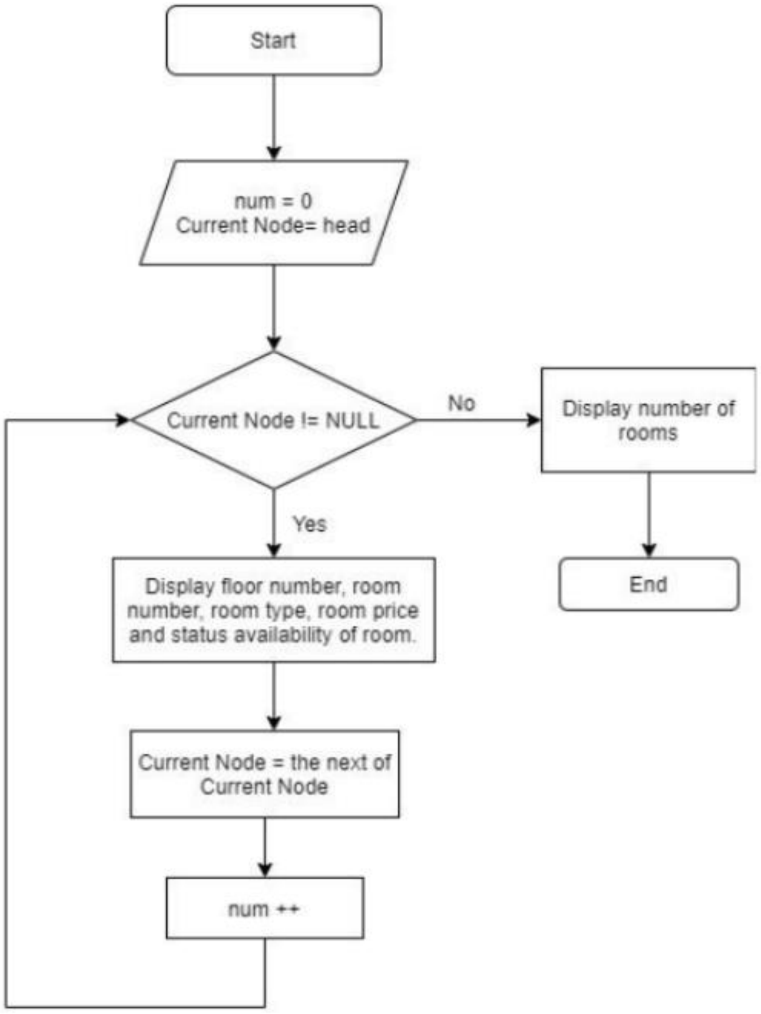

Start
Room Number = n
Current Node = head
Current Index = 1
Find Room. Check whether the room exits in the link list.
Currrent Node && Room Number of the Current Node != n
current node!=NULL
yes
yes
no
Current Node = The next of the current node
result=current node
result =0
Current Index ++
End 1 . Go to 2

Explanation based on flowchart 2:

Delete a room from the system including two parts. First, it checks whether the room exits in the link list. If the first condition is true, it will continue to find the node in the linked list and delete it. For the first part, it will initialize room number as n, current node is pointing to the head of the linked list and current index equal to 1. If the current node and the room number of the current node is not equal to the room number entered by the user, it will continue to loop by pointing the current node to the next of the current node and increase the current index by 1. Once the loop is done, it will check whether the current node is Null or not. If it is null, it will return zero, otherwise, it will return the current node. For the second part, it will use the same theory as explained before to find the node that contains the room number entered by the user and delete the node. After delete the node, it will prompt a message of "You successfully delete the room number(n)". For the fail case (cannot find the corresponding node in the linked list), the system will prompt an error message.

Flow Chart 3: Display room (Customer and Admin)

Data Structure:

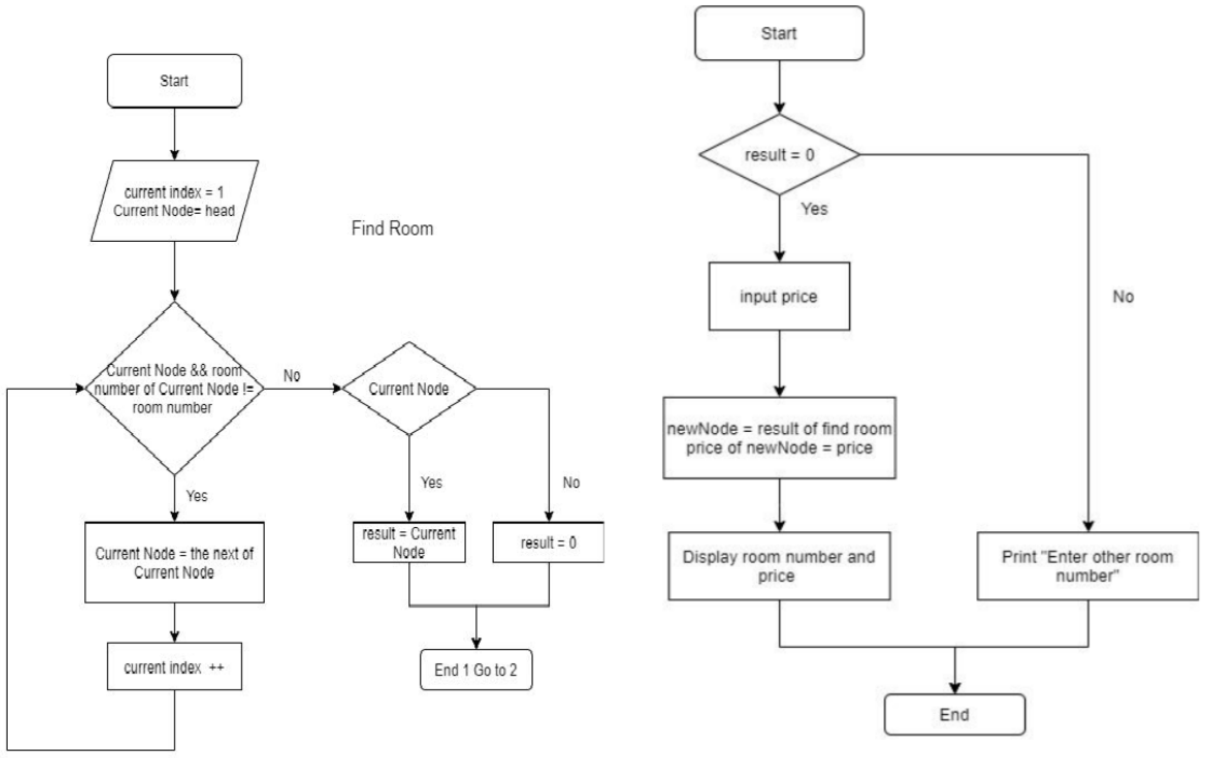

Start
num = 0
Current Node= head
Current Node != NULL
No
Display number of rooms
End
Yes
Display floor number, room number, room type, room price and status availability of room.
Current Node = the next of Current Node
num ++

Explanation based on flowchart 3:

For the function to print the room in the hotel, it will apply the data structure of display in the linked list. The flowchart starts with initializing an integer to zero and pointing the current node to the head of the linked list. The loop will start and continue with the condition that the current node is not null. The loop will display the floor number, room number, room type, room price and status availability of the room. The loop is changing the current node by pointing the current node to the next node in the linked list and increasing the integer by 1. The loop is ended when the condition is false (current node is null)

Flow Chart 4: Change price (Admin)

Data Structure: Delete node and find node in the linked list

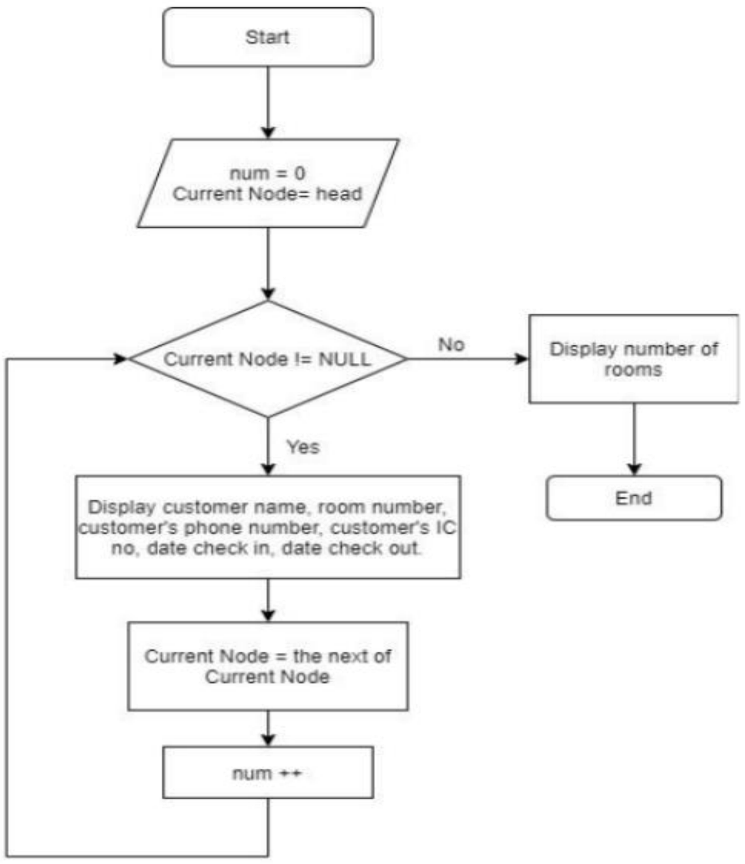


Explanation based on flowchart 4:

For the change in the price of the room by the admin, First, it checks whether the room exists in the linked list. If the first condition is true, it will continue to find the node in the linked list and change the price. For the first part, the current node is pointing to the head of the linked list and current index equal to 1. If the current node and the room number of the current node is not equal to the room number entered by the user, it will continue to loop by pointing the current node to the next of the current node and increase the current index by 1. Once the loop is done, it will check whether it is the current node or not. If it is, it will return zero, otherwise, it will return the current node. For the second part, if the room exists, it will let the admin input the price that wants to change. The newNode points to the result of find room which is the first part then the price of newNode points to the input price. After that, it will display the room number and the changed price. If the room does not exist, it will let the user enter another room number again.

Flow Chart 5: Check customer booking list (Admin)

Data Structure:

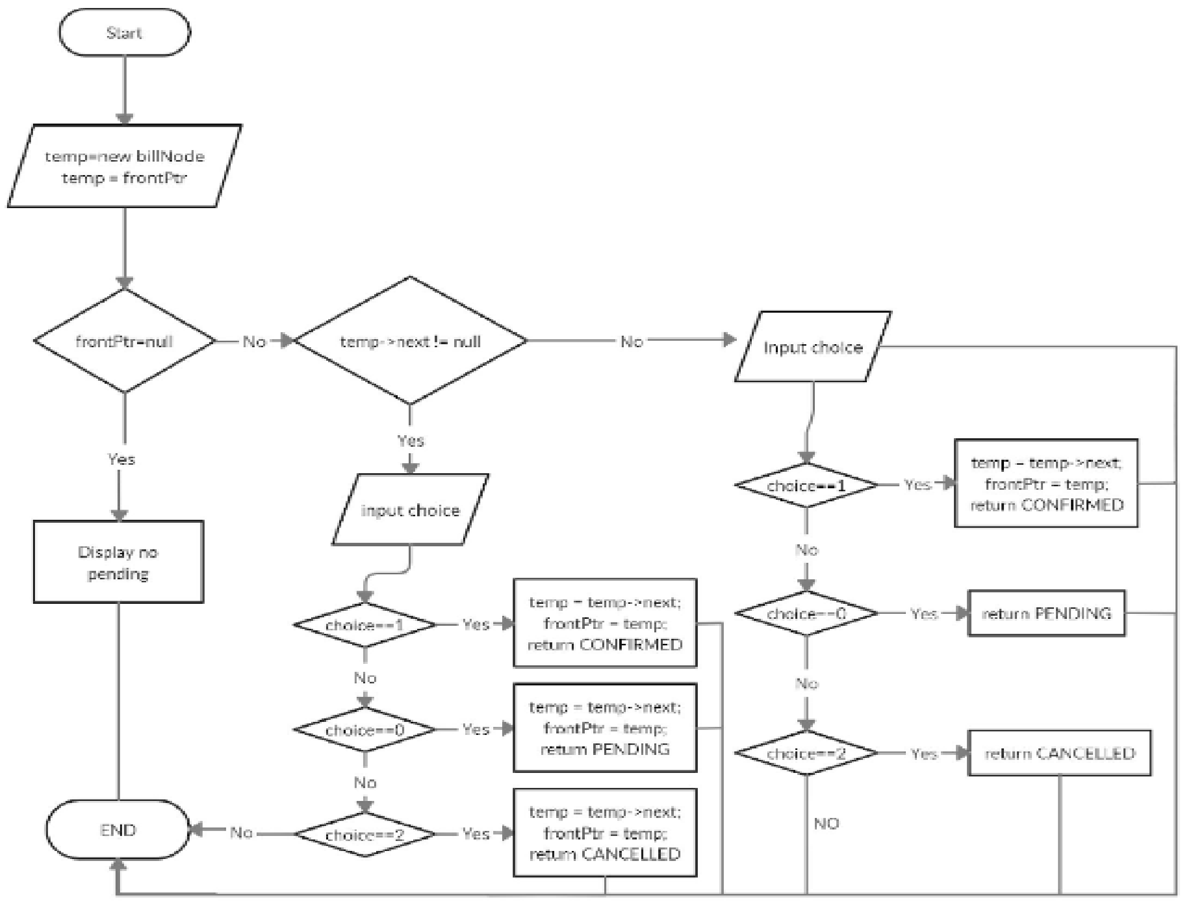

Start
num = 0
Current Node= head
Current Node != NULL
No
Display number of rooms
End
Yes
Display customer name, room number, customer's phone number, customer's IC no, date check in, date check out
Current Node = the next of Current Node
num ++

Explanation for flowchart 5:

Once the customer books a room from the system, it will create a customer linked list to store the information of the customers with their corresponding booked room. Admin have a function of checking the customer booking list in the system. If the customers cancel the room, the customers records and their booked room will be deleted. The flowchart starts to initialize num to zero and points the current node to the head. A loop will be started with a condition that the current node is not null. Inside the loop, it will display the information in the customer node, point the current node to the next node in the linked list, and increment num by 1. The flow chart is ended when the condition of the loop is false and displays the total number of rooms in the system.

Flow Chart 6: Review and confirm pending booking (Admin)

Data Structure: Queue

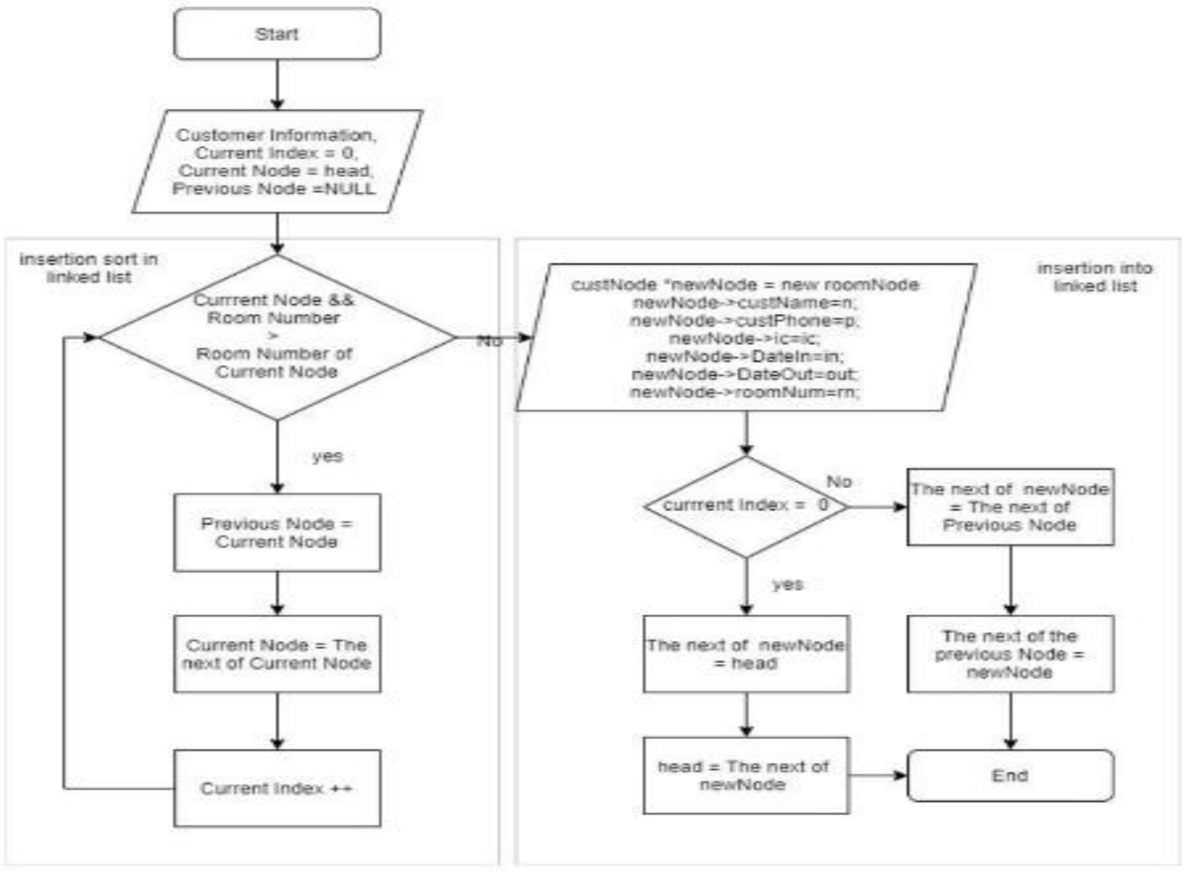


Explanation based on flowchart 6:

The review and confirm pending booking involve the implementation of a queue. The flowchart of this module is mainly about the function of deQueue() that under the class billQueue. The dequeue() function is used to remove and change the status of the rooms in the hotel. The new billNode is dynamically allocated and assigned to pointer temp, the, frontPtr is assigned into location pointed to by pointer temp. First, if the frontPtr is null, it will display to the user that there is no pending booking for review. Else if the next pointer in billNode is not null, the system will request the user to change the status of booking. To confirm the booking, the temp will be assigned to be the next pointer of billNode, and frontPtr will be assigned to be temp which may result in the pending booking to be removed from the queue. Or else if the case does not match with the above condition, the system will request the user to change the status of booking and only the status of the booking that is "CONFIRMED" will be removed from the queue.

Flowchart 7: Book Room

Data Structure: linked list and queue

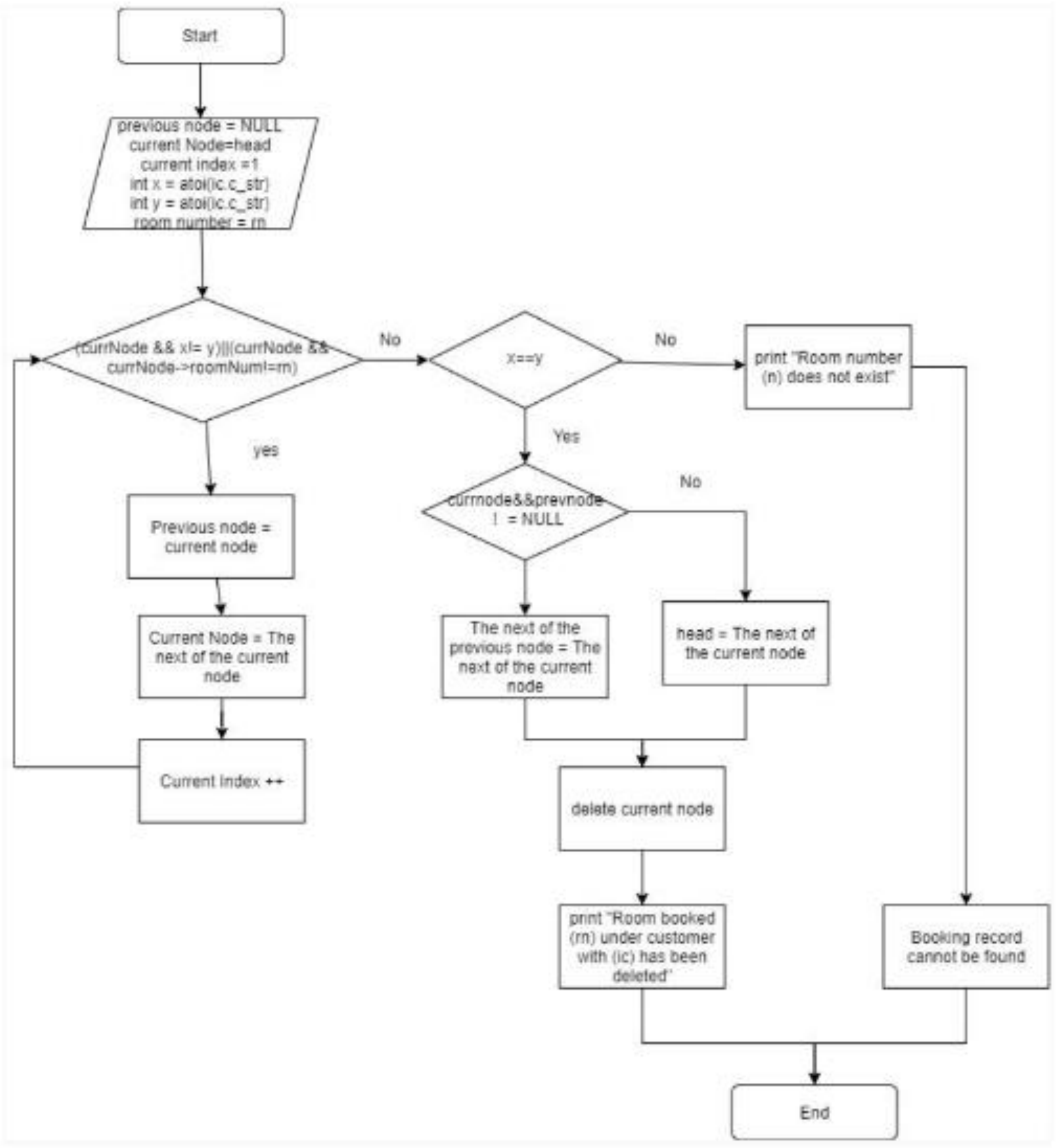

Start
Customer Information, Current Index = 0, Current Node = head, Previous Node =NULL
insertion sort in linked list
Current Node && Room Number > Room Number of Current Node
No
yes
Previous Node = Current Node
Current Node = The next of Current Node
Current Index ++
insertion into linked list
custNode *newNode = new roomNode
newNode->custName=n;
newNode->custPhone=p;
newNode->ic=ic;
newNode->DateIn=in;
newNode->DateOut=out;
newNode->roomNum=rn;
current Index = 0
No
yes
The next of newNode = The next of Previous Node
The next of the previous Node = newNode
The next of newNode = head
head = The next of newNode
End

Explanation based on flowchart 7:

The book room implemented the linked list and queue. For the linked list implementation, first, it checks whether the room number is exits. For the first part, it will initialize the current index equal to 0, current node pointing to the head of linked list and previous node pointing to the NULL of linked list. If the condition of current node and the room number are bigger than the room number of the current node, it will continue looping by pointing previous node is equal to the node and the current node is equal to the next of the current node and last increases the current index by 1.For the second part, if the condition are false, the new node that reference to the custnode is equal to the new room node. It will initialize the new node pointer to custname as n, new node pointer to custphone as p, newnode pointer to ic are ic, newnode pointer to datein as in, newnode pointer to dateout as out an newnode pointer to roomnum as rn. It will continue the loop by pointing that the current index is equal to 0. If this condition are true, it will continue looping by pointing the next of newnode equal to head and the head is equal to the next of newnode and end the booking process. However, it will continue looping by pointing the next of newnode is equal to the next of previous node if the condition is false. Then, it will continue looping by pointing the next of the previous node equal newnode and end the booking process. For the implementation of queue, it initializes the room number as rn, price as p, ic number as ic, number of days as days, datein as in, dateout as out, name as n and the temp reference to billNode equal to new billNode. If the back pointer of the queue equal to NULL, it will continue looping by pointing the temp point to next equal to NULL. Then, user needs to insert the information of the customer into the temp node and front pointer of the queue equal to back pointer of the queue equal to temp. It will end the process or continue regarding the choice of user. However, if the back pointer of the queue is not equal to NULL, the next of the back pointer in the queue equal to temp. Then, user needs to insert the information of the customer into the temp node, the back pointer of queue equal to temp and last ended the flowchart.

Flowchart 8: Cancel Room

Data Structure: Delete node in linked list

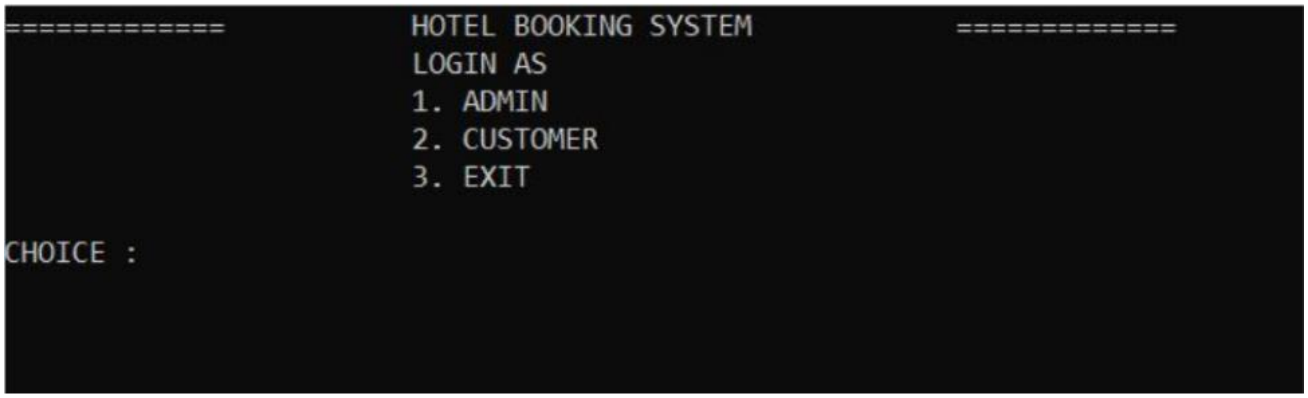

Start
previous node = NULL
current Node=head
current index =1
int x = atoi(ic.c_str)
int y = atoi(ic.c_str)
room number = rn
(currNode && x!= y)||(currNode && currNode->roomNum!=rn)
No
yes
Previous node = current node
Current Node = The next of the current node
Current Index ++
x==y
No
Yes
print "Room number (n) does not exist"
currnode&&prevnode ! = NULL
No
The next of the previous node = The next of the current node
head = The next of the current node
delete current node
print "Room booked (rn) under customer with (ic) has been deleted"
Booking record cannot be found
End

Explanation based on flowchart 8:

For the cancel booking room function, it will first start the find room function and check the room number entered by the user exits in the system and check whether the room is booked. It will only start the flowchart above(delete the customer node in the customer linked list) if the condition(the room exists and is booked) is true. The flowchart will initialize current index as 1 and string rn as the room number. It will point the previous node as NULL, and point the current node to the head of the linked list. It will start a condition to find the position of the finding node by comparing and checking the room number entered by the user. The loop will be continued by pointing previous node to the current node, pointing the current node to the next of the current node and increasing the current index by 1. After the loop is done it will check whether the ic entered by the user is the same with the ic in the found node. If the condition is true, it will start to point the current node to the next of current node(found node in middle or last of the linked list) or point the head to the next of the current node(found node in the first of the linked list) and delete the node. It is done with delete customer nodes in the customer linked list. Lastly, it will change the status of the room node to "available" if the delete customer node process is successful.

PART 3: SYSTEM PROTOTYPE

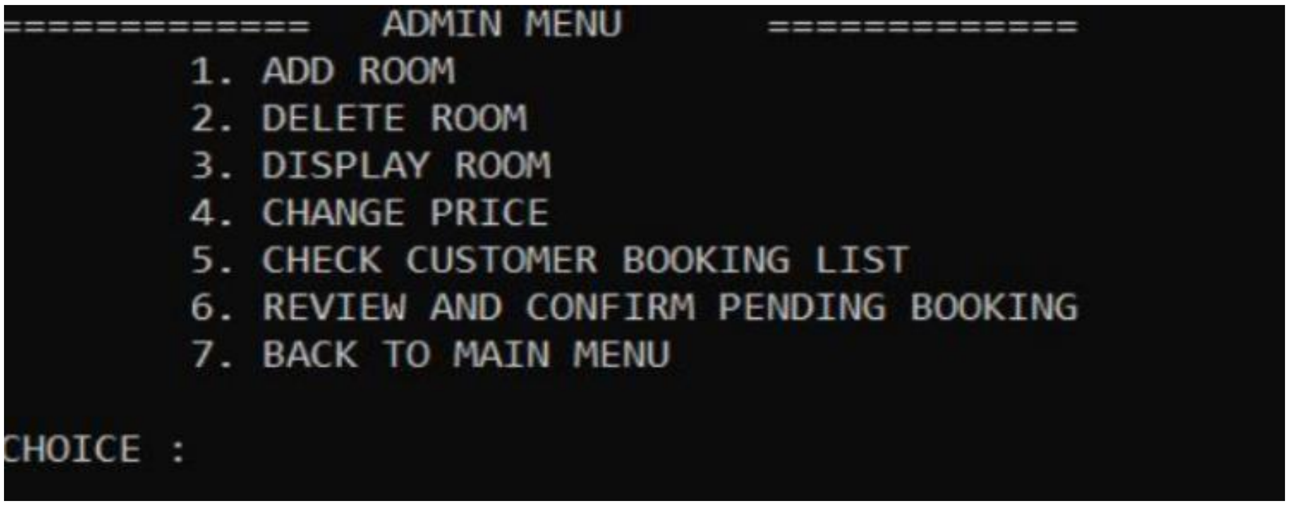
Below are some of the interfaces of our hotel booking system prototype. The users include admin and customer, therefore the interfaces below will be divided based on each user.



```
============            HOTEL BOOKING SYSTEM            ============
                        LOGIN AS
                        1. ADMIN
                        2. CUSTOMER
                        3. EXIT

CHOICE :
```

Screen 1: Hotel Booking menu

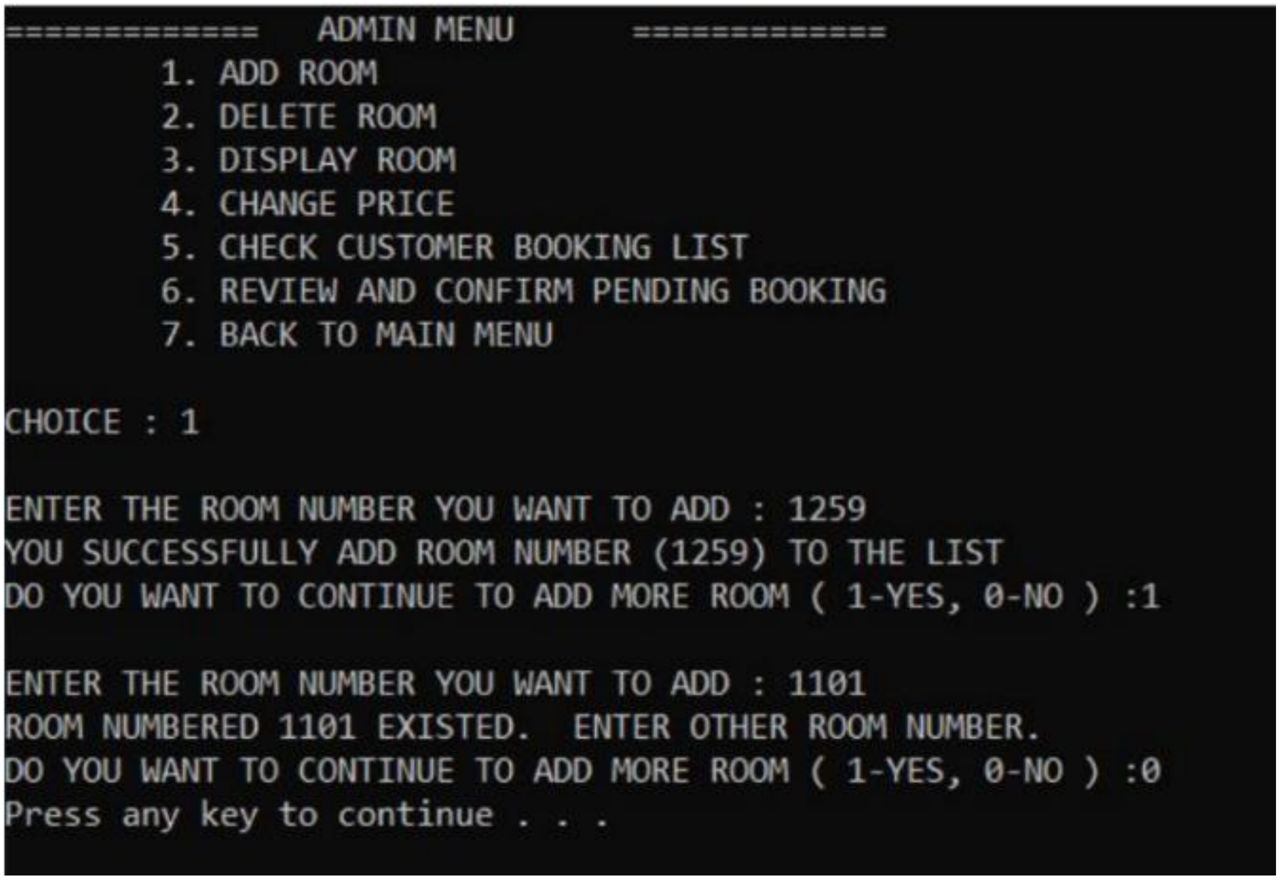
Screen 1 : The user needs to enter the integer value in the range of 1 to 3.If the user enter 1 , user will login as an admin, and if user enter 2 will login as a customer and if user enter 3 it will exit the program. Otherwise, the program will prompt invalid choice and the screen will display again after the user presses any key



```
=============   ADMIN MENU      =============
        1. ADD ROOM
        2. DELETE ROOM
        3. DISPLAY ROOM
        4. CHANGE PRICE
        5. CHECK CUSTOMER BOOKING LIST
        6. REVIEW AND CONFIRM PENDING BOOKING
        7. BACK TO MAIN MENU

CHOICE :
```

Screen 2: Admin Menu

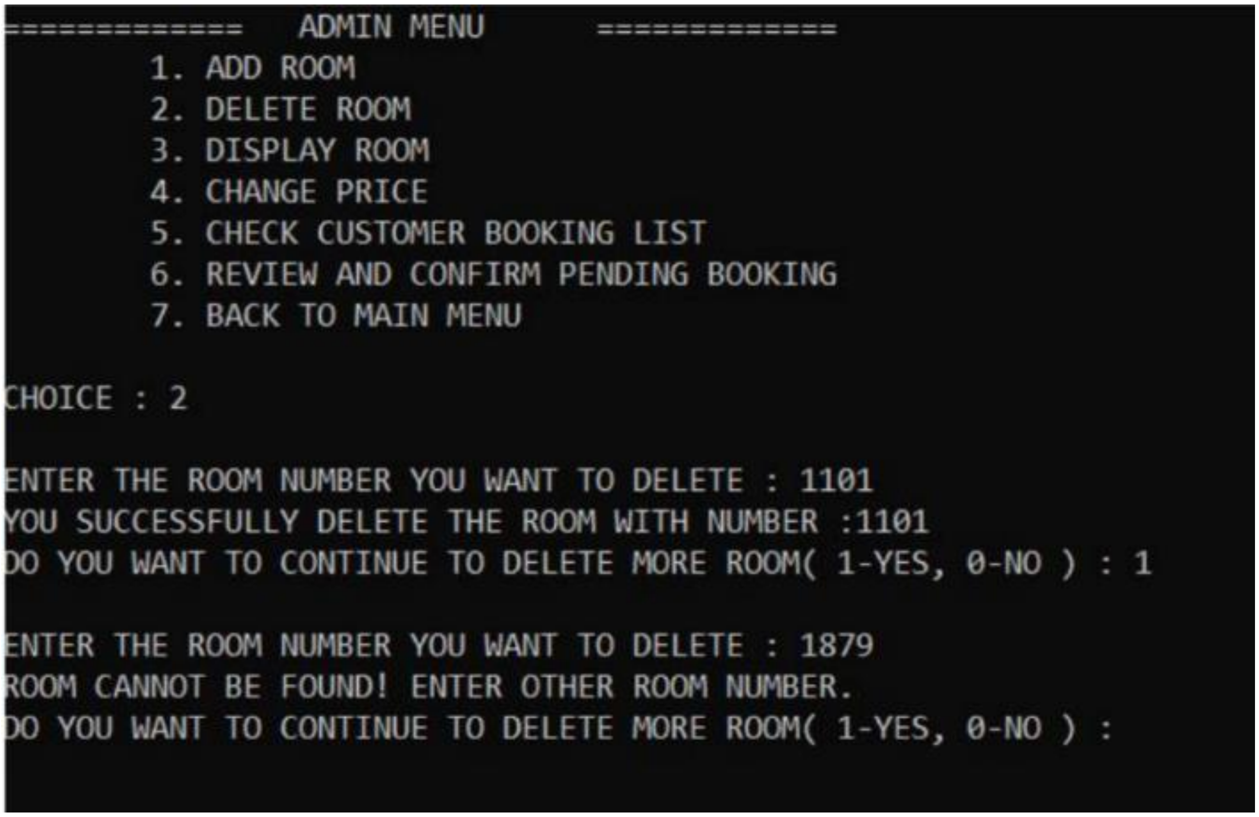
Screen 2: If the user enters integer 1 after choosing admin, it will display an admin menu. The user needs to enter the integer value in the range of 1 to 7 according to what choices they want. If the user enters the other number, the system will prompt invalid choice and the screen will display again after the user presses any key.



```
=============   ADMIN MENU      =============
        1. ADD ROOM
        2. DELETE ROOM
        3. DISPLAY ROOM
        4. CHANGE PRICE
        5. CHECK CUSTOMER BOOKING LIST
        6. REVIEW AND CONFIRM PENDING BOOKING
        7. BACK TO MAIN MENU

CHOICE : 1

ENTER THE ROOM NUMBER YOU WANT TO ADD : 1259
YOU SUCCESSFULLY ADD ROOM NUMBER (1259) TO THE LIST
DO YOU WANT TO CONTINUE TO ADD MORE ROOM ( 1-YES, 0-NO ) :1

ENTER THE ROOM NUMBER YOU WANT TO ADD : 1101
ROOM NUMBERED 1101 EXISTED.  ENTER OTHER ROOM NUMBER.
DO YOU WANT TO CONTINUE TO ADD MORE ROOM ( 1-YES, 0-NO ) :0
Press any key to continue . . .
```

Screen 3: Admin - Adding room

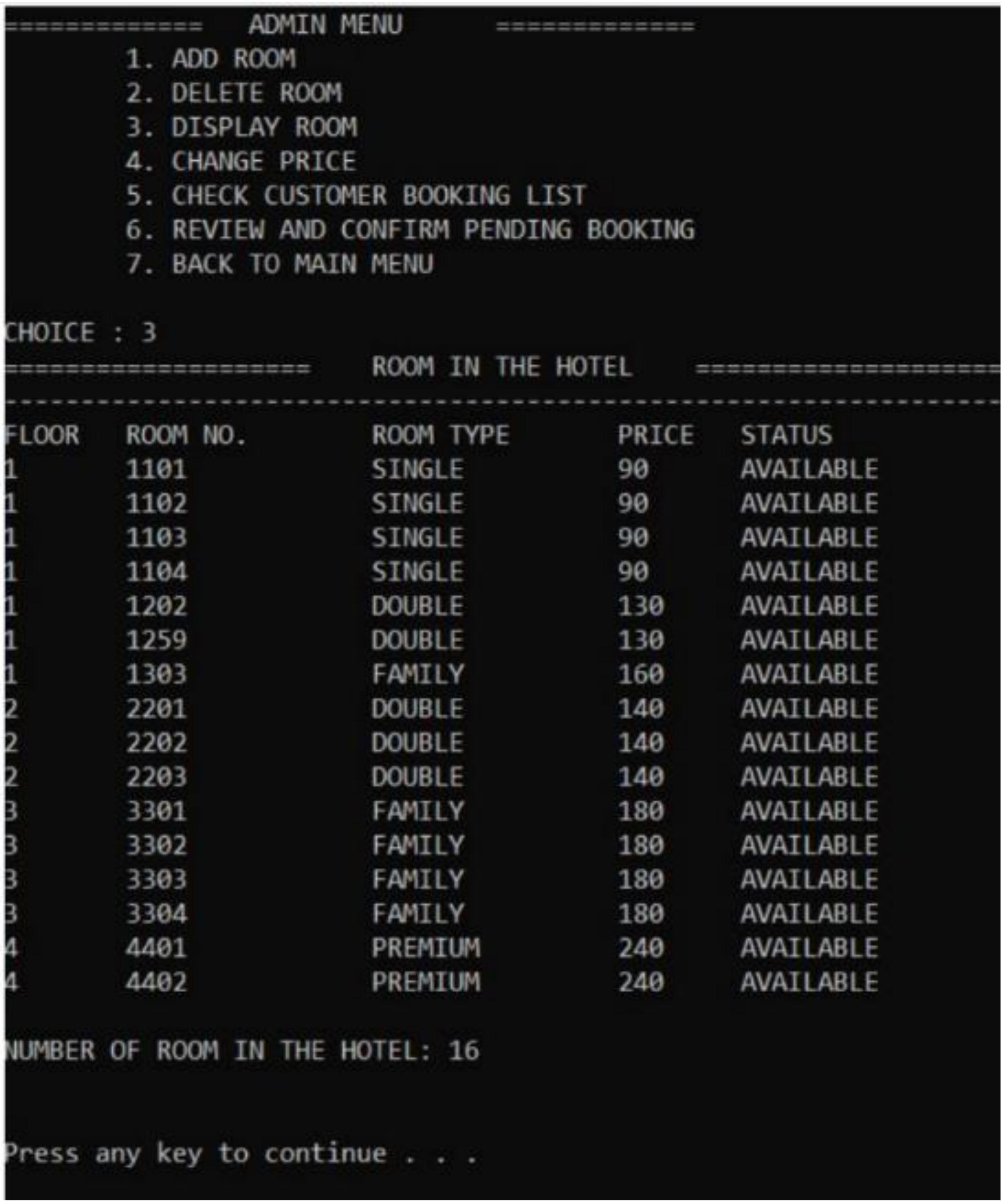
Screen 3: For choice 1, if the user enters a new room number, it will display a successful message. If the user enters an existing room number, it will display an error message and request the user to try another room number. After that, it will display an option to have or not for continuing the adding room process. If wanted, enter 1 else enter 0 if not interested to continue adding. The screen of the admin menu will display again after the user presses any key.



```
============   ADMIN MENU      ============
       1. ADD ROOM
       2. DELETE ROOM
       3. DISPLAY ROOM
       4. CHANGE PRICE
       5. CHECK CUSTOMER BOOKING LIST
       6. REVIEW AND CONFIRM PENDING BOOKING
       7. BACK TO MAIN MENU

CHOICE : 2

ENTER THE ROOM NUMBER YOU WANT TO DELETE : 1101
YOU SUCCESSFULLY DELETE THE ROOM WITH NUMBER :1101
DO YOU WANT TO CONTINUE TO DELETE MORE ROOM( 1-YES, 0-NO ) : 1

ENTER THE ROOM NUMBER YOU WANT TO DELETE : 1879
ROOM CANNOT BE FOUND! ENTER OTHER ROOM NUMBER.
DO YOU WANT TO CONTINUE TO DELETE MORE ROOM( 1-YES, 0-NO ) :
```

Screen 4: Admin - Deleting room

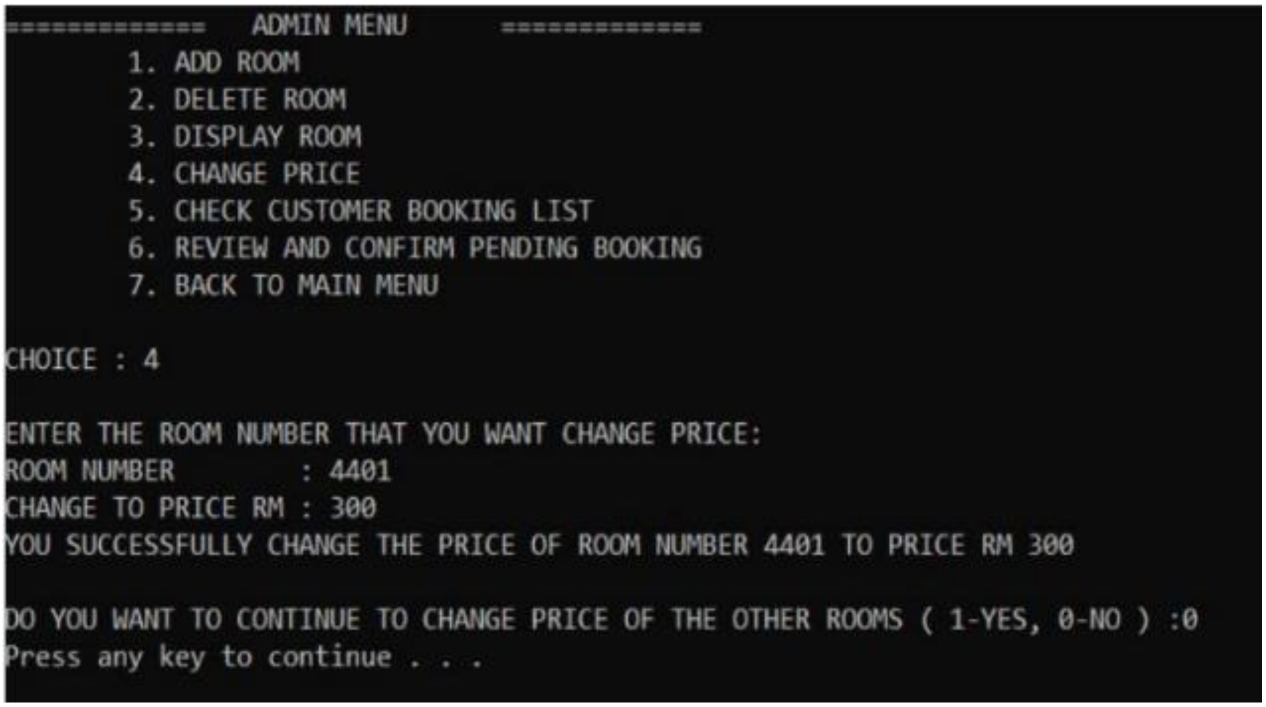
Screen 4: For choice 2, it will allow the user to enter a room number to delete a room. If the user enters an existing room number, it will display a successful message or else it will request the user to enter another room number. After that, it will display an option to have or not for continuing the deleting room process. If wanted, enter 1 else enter 0 if not interested to continue deleting. The screen of the admin menu will display again after the user presses any key.



```
=============   ADMIN MENU      =============
        1. ADD ROOM
        2. DELETE ROOM
        3. DISPLAY ROOM
        4. CHANGE PRICE
        5. CHECK CUSTOMER BOOKING LIST
        6. REVIEW AND CONFIRM PENDING BOOKING
        7. BACK TO MAIN MENU

CHOICE : 3
====================    ROOM IN THE HOTEL    ====================
-----------------------------------------------------------------
FLOOR   ROOM NO.        ROOM TYPE       PRICE   STATUS
1       1101            SINGLE          90      AVAILABLE
1       1102            SINGLE          90      AVAILABLE
1       1103            SINGLE          90      AVAILABLE
1       1104            SINGLE          90      AVAILABLE
1       1202            DOUBLE          130     AVAILABLE
1       1259            DOUBLE          130     AVAILABLE
1       1303            FAMILY          160     AVAILABLE
2       2201            DOUBLE          140     AVAILABLE
2       2202            DOUBLE          140     AVAILABLE
2       2203            DOUBLE          140     AVAILABLE
3       3301            FAMILY          180     AVAILABLE
3       3302            FAMILY          180     AVAILABLE
3       3303            FAMILY          180     AVAILABLE
3       3304            FAMILY          180     AVAILABLE
4       4401            PREMIUM         240     AVAILABLE
4       4402            PREMIUM         240     AVAILABLE

NUMBER OF ROOM IN THE HOTEL: 16


Press any key to continue . . .
```

Screen 5: Admin - Displaying rooms in the hotel

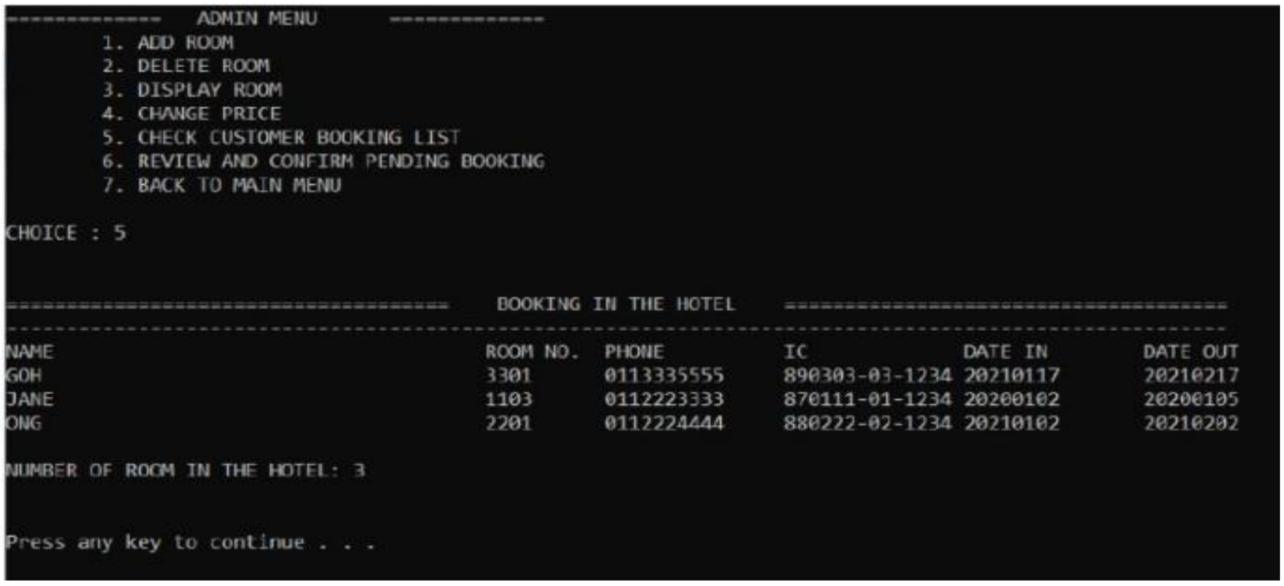
Screen 5: For choice 3 of admin, it will display the rooms in the hotel and their status. The details of the room in the hotel included floor, room number, room type, price and status of availability. If the user enters any key, the system will prompt to the admin menu.



```
=============    ADMIN MENU      =============
        1. ADD ROOM
        2. DELETE ROOM
        3. DISPLAY ROOM
        4. CHANGE PRICE
        5. CHECK CUSTOMER BOOKING LIST
        6. REVIEW AND CONFIRM PENDING BOOKING
        7. BACK TO MAIN MENU

CHOICE : 4

ENTER THE ROOM NUMBER THAT YOU WANT CHANGE PRICE:
ROOM NUMBER        : 4401
CHANGE TO PRICE RM : 300
YOU SUCCESSFULLY CHANGE THE PRICE OF ROOM NUMBER 4401 TO PRICE RM 300

DO YOU WANT TO CONTINUE TO CHANGE PRICE OF THE OTHER ROOMS ( 1-YES, 0-NO ) :0
Press any key to continue . . .
```

Screen 6: Admin - Change price of the rooms

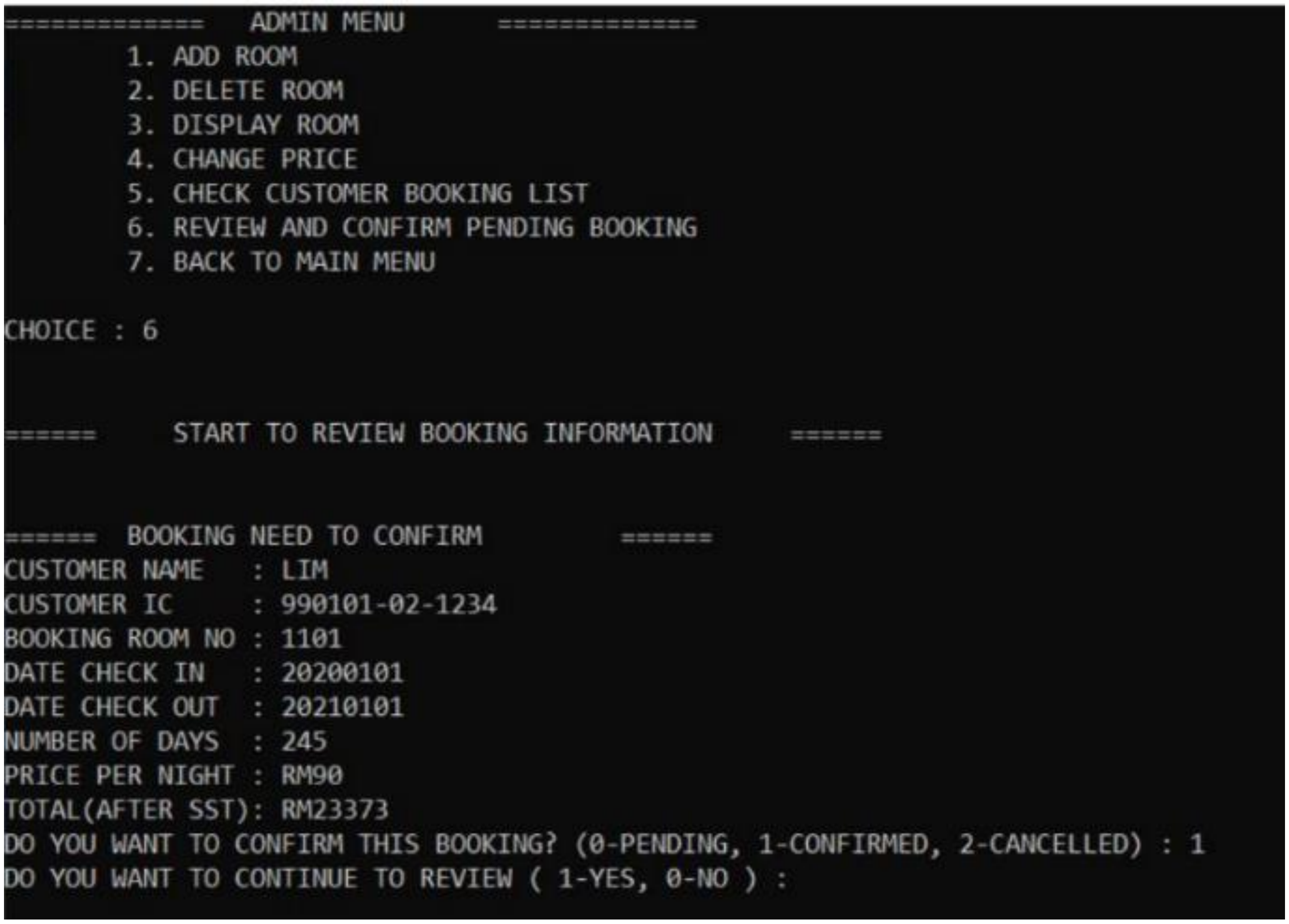
Screen 6: User needs to insert a room number that wants to change price. After that, the user needs to insert a new price for the related room number. Then, the system will display a successful message on the screen. Users need to enter 0 or 1 to continue or not for changing the room price.



```
============    ADMIN MENU      ============
        1. ADD ROOM
        2. DELETE ROOM
        3. DISPLAY ROOM
        4. CHANGE PRICE
        5. CHECK CUSTOMER BOOKING LIST
        6. REVIEW AND CONFIRM PENDING BOOKING
        7. BACK TO MAIN MENU

CHOICE : 5


====================================    BOOKING IN THE HOTEL    ====================================
-----------------------------------------------------------------------------------------------------
NAME                                    ROOM NO.  PHONE          IC                DATE IN        DATE OUT
GOH                                     3301      0113335555     890303-03-1234 20210117       20210217
JANE                                    1103      0112223333     870111-01-1234 20200102       20200105
ONG                                     2201      0112224444     880222-02-1234 20210102       20210202

NUMBER OF ROOM IN THE HOTEL: 3


Press any key to continue . . .
```

Screen 7: Admin - Checking booking list

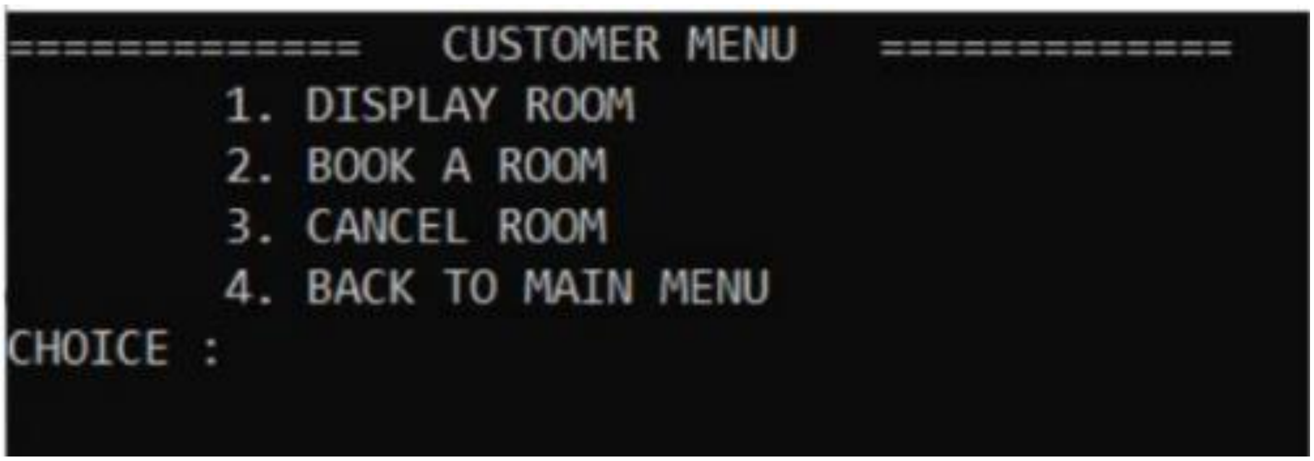
Screen 7: The system will show all the booking details of the customer. Admin able to check with the existing booking and the customer details. Users need to enter any key to continue.



```
=============    ADMIN MENU      =============
        1. ADD ROOM
        2. DELETE ROOM
        3. DISPLAY ROOM
        4. CHANGE PRICE
        5. CHECK CUSTOMER BOOKING LIST
        6. REVIEW AND CONFIRM PENDING BOOKING
        7. BACK TO MAIN MENU

CHOICE : 6


======     START TO REVIEW BOOKING INFORMATION     ======


======  BOOKING NEED TO CONFIRM         ======
CUSTOMER NAME   : LIM
CUSTOMER IC     : 990101-02-1234
BOOKING ROOM NO : 1101
DATE CHECK IN   : 20200101
DATE CHECK OUT  : 20210101
NUMBER OF DAYS  : 245
PRICE PER NIGHT : RM90
TOTAL(AFTER SST): RM23373
DO YOU WANT TO CONFIRM THIS BOOKING? (0-PENDING, 1-CONFIRMED, 2-CANCELLED) : 1
DO YOU WANT TO CONTINUE TO REVIEW ( 1-YES, 0-NO ) :
```

Screen 8: Admin - Reviewing and confirm booking

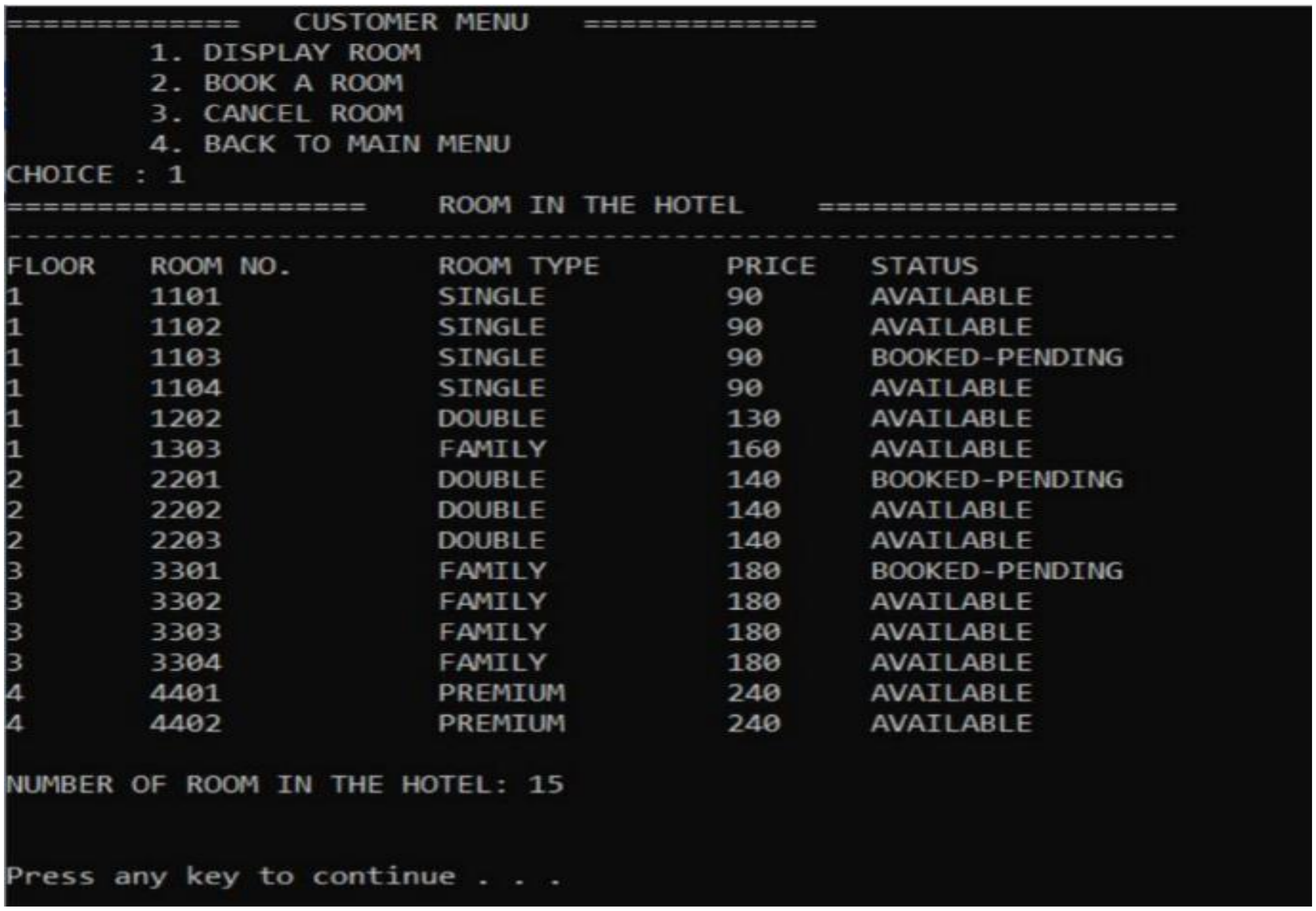
Screen 8: The system will show all the bookings that need to be confirmed. To change the status of booking, the user needs to enter the integer value in the range of 0 to 2 where 0 is pending, 1 is confirmed and 2 is cancelled.



```
=============   CUSTOMER MENU   =============
         1. DISPLAY ROOM
         2. BOOK A ROOM
         3. CANCEL ROOM
         4. BACK TO MAIN MENU
CHOICE :
```

Screen 9: Customer Menu

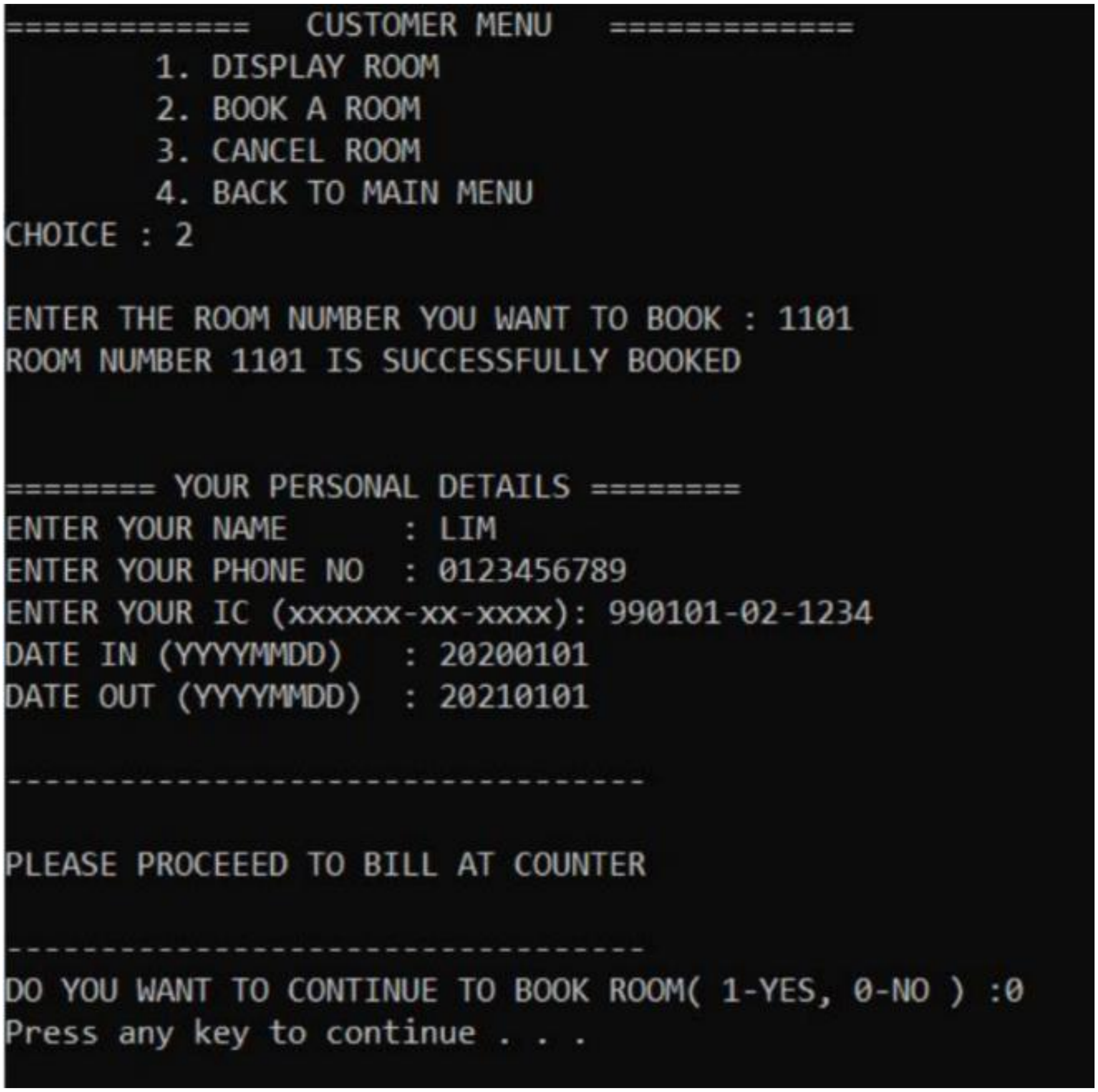
Screen 9: The user needs to enter the integer value in the range of 1 to 4 according to what choices they want. If the user enters the other number, the system will prompt invalid choice and the screen will display again after the user presses any key.



```
=============   CUSTOMER MENU   =============
        1. DISPLAY ROOM
        2. BOOK A ROOM
        3. CANCEL ROOM
        4. BACK TO MAIN MENU
CHOICE : 1
====================    ROOM IN THE HOTEL    ====================
-----------------------------------------------------------------
FLOOR   ROOM NO.        ROOM TYPE       PRICE   STATUS
1       1101            SINGLE          90      AVAILABLE
1       1102            SINGLE          90      AVAILABLE
1       1103            SINGLE          90      BOOKED-PENDING
1       1104            SINGLE          90      AVAILABLE
1       1202            DOUBLE          130     AVAILABLE
1       1303            FAMILY          160     AVAILABLE
2       2201            DOUBLE          140     BOOKED-PENDING
2       2202            DOUBLE          140     AVAILABLE
2       2203            DOUBLE          140     AVAILABLE
3       3301            FAMILY          180     BOOKED-PENDING
3       3302            FAMILY          180     AVAILABLE
3       3303            FAMILY          180     AVAILABLE
3       3304            FAMILY          180     AVAILABLE
4       4401            PREMIUM         240     AVAILABLE
4       4402            PREMIUM         240     AVAILABLE

NUMBER OF ROOM IN THE HOTEL: 15


Press any key to continue . . .
```

Screen 10: Customer - Displaying room in the hotel

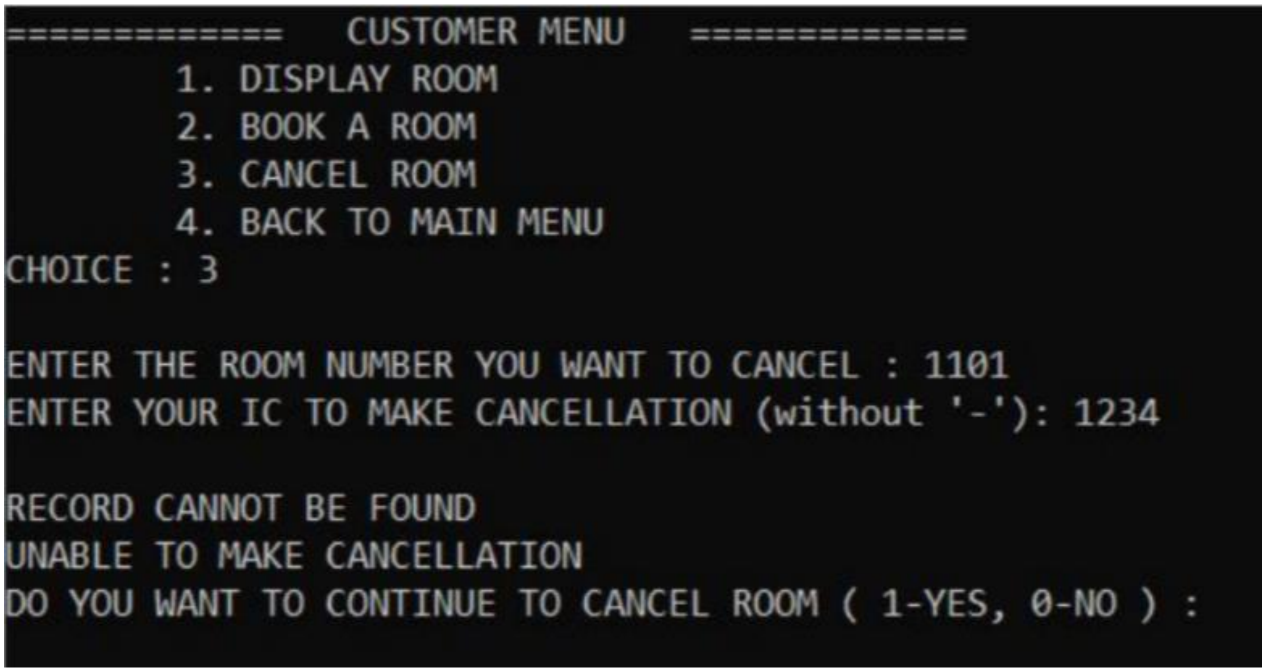
Screen 10: It will display all the rooms in the hotel. The details of the room in the hotel included floor, room number, room type, price and status. If the user enters any key, the system will prompt to the customer menu.



```
============   CUSTOMER MENU   ============
        1. DISPLAY ROOM
        2. BOOK A ROOM
        3. CANCEL ROOM
        4. BACK TO MAIN MENU
CHOICE : 2

ENTER THE ROOM NUMBER YOU WANT TO BOOK : 1101
ROOM NUMBER 1101 IS SUCCESSFULLY BOOKED


======== YOUR PERSONAL DETAILS ========
ENTER YOUR NAME      : LIM
ENTER YOUR PHONE NO  : 0123456789
ENTER YOUR IC (xxxxxx-xx-xxxx): 990101-02-1234
DATE IN (YYYYMMDD)   : 20200101
DATE OUT (YYYYMMDD)  : 20210101

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

PLEASE PROCEEED TO BILL AT COUNTER

----------------------------------
DO YOU WANT TO CONTINUE TO BOOK ROOM( 1-YES, 0-NO ) :0
Press any key to continue . . .
```

Screen 11: Customer - Booking a room

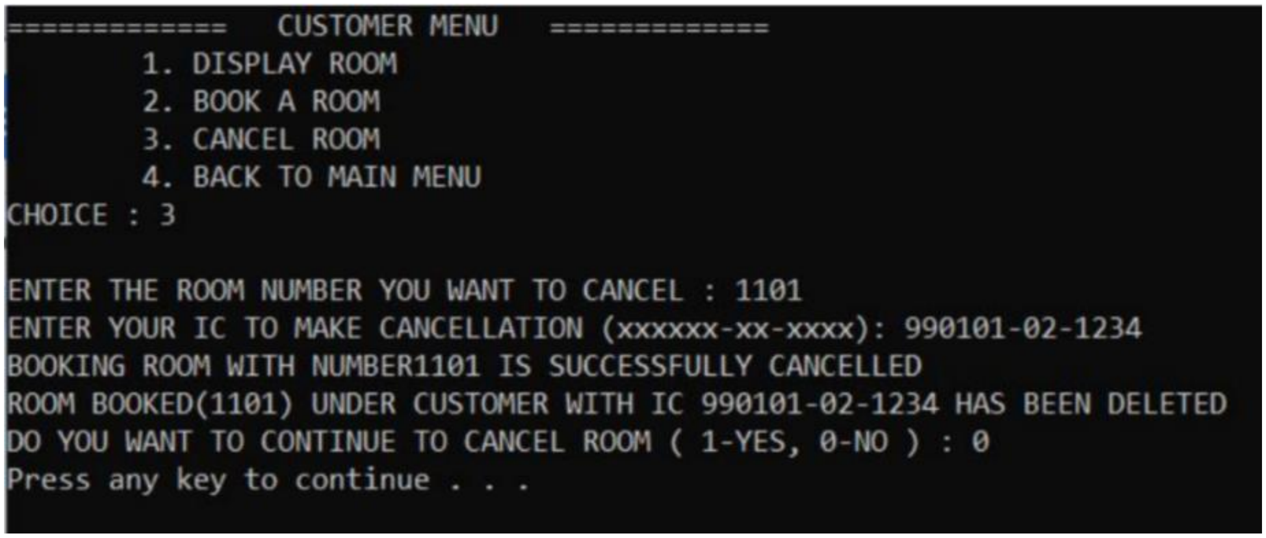
Screen 11: It will display a successfully booked message if the user entered the matched room number in the hotel. After that, users need to enter their personality such as their name, phone number, number of ic, date check in and date check out with the correct format. Then, it will display “please proceed to the bill at the counter”. Users will then need to enter 1 or 0 to continue or not for their booking.



```
=============   CUSTOMER MENU   =============
        1. DISPLAY ROOM
        2. BOOK A ROOM
        3. CANCEL ROOM
        4. BACK TO MAIN MENU
CHOICE : 3

ENTER THE ROOM NUMBER YOU WANT TO CANCEL : 1101
ENTER YOUR IC TO MAKE CANCELLATION (without '-'): 1234

RECORD CANNOT BE FOUND
UNABLE TO MAKE CANCELLATION
DO YOU WANT TO CONTINUE TO CANCEL ROOM ( 1-YES, 0-NO ) :
```

Screen 12: Customer UNABLE cancelling the room

Screen 12: If the user enters the room that wishes to cancel is invalid, then the system will display a record not found and unable to make cancellation. After that, the user needs to enter 1 for continuing or 0 quit the cancellation process.



```
============   CUSTOMER MENU   ============
        1. DISPLAY ROOM
        2. BOOK A ROOM
        3. CANCEL ROOM
        4. BACK TO MAIN MENU
CHOICE : 3

ENTER THE ROOM NUMBER YOU WANT TO CANCEL : 1101
ENTER YOUR IC TO MAKE CANCELLATION (xxxxxx-xx-xxxx): 990101-02-1234
BOOKING ROOM WITH NUMBER1101 IS SUCCESSFULLY CANCELLED
ROOM BOOKED(1101) UNDER CUSTOMER WITH IC 990101-02-1234 HAS BEEN DELETED
DO YOU WANT TO CONTINUE TO CANCEL ROOM ( 1-YES, 0-NO ) : 0
Press any key to continue . . .
```

Screen 13: Customer SUCCESSFULLY cancelling the room

Screen 13: User needs to enter the room that wishes to cancel is valid, then the system will display successfully cancelled. After that, the user needs to enter 1 to continue or 0 to stop the cancellation process.

PART 4: APPENDIX HARDCOPY OF SOURCE CODE

```
#include <iostream>
#include <iomanip>
#include <string>
#include <conio.h>
#include <stdlib.h>
#include <cmath>
#include <cctype>

using namespace std;
class billNode{
public:
 int roomNum;
float price;
string ic;
string name;
double total;
 string DateIn;
 string DateOut;
int days;
billNode *next;
};
class billQueue{
billNode *frontPtr, *backPtr;
public:
billQueue(){
backPtr=NULL;
frontPtr=NULL;
}
~billQueue(){
billNode *temp =frontPtr;
while(temp){
        frontPtr=temp->next;
        delete temp;
        temp=frontPtr;
}
}
```

```
bool isEmpty(){
        return (backPtr==NULL&&frontPtr==NULL);
}
// Implementation of queue
void enqueue(int rn, float p, string ic_, string n, string in, string out, int
d){
    billNode * temp=new billNode;
    if(backPtr==NULL){
        temp->next=NULL;
        temp->roomNum=rn;
        temp->price=p;
        temp->ic=ic_;
        temp->name=n;
        temp->DateIn=in;
        temp->DateOut=out;
        temp->days=d;
        backPtr=frontPtr=temp;
 }
 else{
        backPtr->next=temp;
        temp->roomNum=rn;
        temp->price=p;
        temp->ic=ic_;
        temp->name=n;
        temp->DateIn=in;
        temp->DateOut=out;
        temp->days=d;
        temp->next=NULL;
        backPtr=temp;
 }
}
void displayInvoice(billNode * temp){
        if(temp->days<=1){
                temp->days==1;
        }
        temp->total=temp->price*1.06*(temp->days*1.0);
        cout << "\n===========BOOKING NEED TO CONFIRM===============""<<endl;
        cout << "\n CUSTOMER NAME\t : "<< temp->name<<endl;
        cout << "\n ROOM NUMBER \t: "<< temp->roomNum<<endl;
        cout << "\n CHECK IN DATE:\t"<< temp->DateIn<<endl;
        cout << "\n CHECK OUT DATE:\t"<< temp->DateOut<<endl;
        cout << "\n DAYS STAYED:\t"<< temp->days<<endl;
        cout << "\n TOTAL AMOUNT:\t"<< temp->total<<endl;
        cout << "\nCUSTOMER IC:\t"<<temp->ic<<endl;
}
```

```
// Implimentation of queue
string dequeue(){
        billNode * temp = new billNode;
        temp= frontPtr;
        if (frontPtr==NULL){
                cout << "No pending booking need to be confirmed "<<endl;
                return "NULL";
        }
        else if(temp->next!=NULL){
                int choice;
                displayInvoice(temp);
                do{
                        cout <<"\n Do you want to confirm your booking? (0-
Pending, 1-Confirmed, 2-Cancelled):"<<endl;
                        cin >> choice;
                        if (choice==1){
                                temp=temp->next;
                                frontPtr=temp;
                                return "Booking Confirmed";
                        }
                        else if (choice==0){
                                temp= temp-> next;
                                frontPtr=temp;
                                 return "Booking Pending ";
                        }
                        else if(choice ==2){
                           temp=temp->next;
                                frontPtr=temp;
                                return "Booking Cancelled";
                        }

                }while(choice<0&&choice>2);
        }
}
billNode *getFrontRN(){
        billNode * t =new billNode;
        if (frontPtr!=NULL){
                t=frontPtr;
        }
        return t;
}
};
class custNode{
        public:
        string custName;
```

```
         string custPhone;
          string ic;
           string DateIn;
            string DateOut;
            int roomNum;
            custNode *next;

};
class customer{
        custNode *head;
        public:
        customer()
        {head=NULL;}
void sortedCustIn(string n, string p, string ic,  string in,  string out,int rn){
        int currIndex=0;
        custNode *currNode=head;
        custNode *prevNode=NULL;
        while(currNode&& n>currNode->custName){
                prevNode=currNode;
                currNode=currNode->next;
                currIndex++;
        }
        custNode* newNode=new custNode();
        newNode->custName=n;
        newNode->custPhone=p;
        newNode->ic=ic;
        newNode->DateIn=in;
        newNode->DateOut=out;
        newNode->roomNum=rn;
        if(currIndex==0){
                newNode->next=head;
                head=newNode;
                }else{
                        newNode->next=prevNode->next;
                        prevNode->next=newNode;
        }
       }
       void deleteCust( string ic, int rn){
        custNode *currNode=head;
        custNode *prevNode=NULL;
        int currIndex=1;
        string IC= currNode->ic.substr(0,12);
        int y=atoi(IC.c_str());
        int x=atoi(ic.c_str());
        while((currNode&&x!=y)||(currNode&&currNode->roomNum!=rn)){
                prevNode=currNode;
                currNode=currNode->next;
                currIndex++;
        }
        if (x==y){
                if(currNode){
                        if(prevNode){
                                prevNode->next=currNode->next;
                                delete currNode;
                        }
                        else {
                                head=currNode->next;
                                delete currNode;
                        }
                }
                 cout << "ROOM BOOKED("<<rn<<") UNDER CUSTOMER WITH IC "<<ic<<"
HAS BEEN DELETED"<<endl;
        }
       else{
        cout << "BOOKING RECORD OF"<<rn<<" UNDER CUSTOMER WITH IC "<<ic<<" I CAN
NOT FOUND" <<endl;
       }

       }
       bool findCust(string _ic){
        custNode *currNode = head;
        int currIndex=1;
        string IC= currNode->ic.substr(0,13);
        int y=atoi(IC.c_str());
        int x=atoi(_ic.c_str());
        if (x==y){
                return true;
        }
        else
        return false;
       }
       void displayCustRoom(){
        int num=0;
        custNode* currNode = head;
        cout<<"|-------------------BOOKING IN HOTEL--------------------------
|"<<endl;
        cout<<left<<setw(40)<<"NAME"<<setw(11)<<"ROOM"<<setw(15)<<"PHONE"<<setw(2
0)<< "IC"<<setw(12)<<"DATE IN"<< setw(12)<<"DATE OUT"<<endl;
        while(currNode != NULL){
               cout <<left<<setw(40)<<currNode->custName<<setw(11)<<currNode-
>roomNum<<setw(15)<<currNode->custPhone<<setw(20)<<currNode-
>ic<<setw(12)<<currNode->DateIn<<setw(12)<<currNode->DateOut<<endl;
            currNode= currNode->next;
                num++;
        }
        cout<<"\n NUMBER OF ROOM IN HOTEL :"<< num<<endl;
       }
};
class roomNode{
        public:
        int floorNo;
        int roomNum;
        int roomType;
        float price;
        int status;
        roomNode *next;
        roomNode *prev;
};
string rt(int rt){
        if(rt==1){
                return "SINGLE";
        }
        else if(rt==2){
                return "Double";
        }
        else if(rt==3){
                return "FAMILY";
        }
        else if (rt==4){
                return "PREMIUM";
        }
        else
        return "DEFAULT";
}
string status(int status){
        if( status==1){
                return "BOOKED CONFIRMED";
        }
       else if(status==2){
          return "AVAILABLE";
        }
        else {
                return "DEFAULT";
        }
```

```
}
class room{
private:
roomNode *head;
public:
room(){
        head=NULL;
}
void insertSortedRoom( int roomNum){

        int currIndex=0;
        roomNode *currNode=head;
        roomNode *prevNode=NULL;
        while(currNode&&roomNum>currNode->roomNum){
            prevNode=currNode;
            currNode=currNode->next;
            currIndex++;
        }
        roomNode *newNode=new roomNode;
        newNode->roomNum=roomNum;
        newNode->floorNo=roomNum%1000;
        newNode->roomType=(roomNum-(roomNum%1000*1000))/100;
        newNode->price=setprice(newNode);
        newNode->status=0;
        if(currIndex==0){
                newNode->next=head;
                head=newNode;
        }
        else{
                newNode->next=prevNode->next;
                prevNode->next=newNode;
        }
}
float setprice(roomNode *n){
        float p=0;
        roomNode *newNode=new roomNode;
        newNode =n;
        if(newNode->roomType==1){
                p+=80.00;
        }
        else if(newNode ->roomType==2){
                p+=120.00;
        }
        else if(newNode ->roomType==3){
                p+=150.00;
        }
        else if(newNode ->roomType==4){
                p+=200.00;
        }
        else {
                p=0.00;
        }
        p+=(newNode->floorNo*10.00);
        return p;
}
void deleteRoom(int rn){
        roomNode *prevNode=NULL;
        roomNode *currNode=head;
        int currIndex=1;
        while(currNode&&currNode->roomNum!=rn){
                prevNode=currNode;
                currNode=currNode->next;
                currIndex++;
        }
        if( currNode){
                if(prevNode){
                        prevNode->next=currNode->next;
                        delete currNode;
                }
                else {
                        head=currNode->next;
                        delete currNode;
                }
                cout << "YOU SUCCESSFULLY DELETE THE ROOM WITH NUMBER
:"<<rn<<endl;
        }
        else {
                cout << "ROOM NUMBER ("<<rn<<") DOES NOT EXIST "<<endl;
        }
}
bool bookRoom(int rn){
        roomNode *newNode=new roomNode;
        newNode=findRoom(rn);
        if(!newNode){
                cout << "ROOM NUMBER "<<rn<<"CAN NOT BE FOUND\n"<<endl;
                return false;
        }
        else {
                if( newNode->status==0){
                        newNode->status=1;
```

```
                         cout << "ROOM NUMBER "<<rn<<"IS SUCCESSFULLY 
BOOKED\n"<<endl;
                return true;
                }
                else{
                         cout << "ROOM NUMBER "<<rn<<"IS NOT AVAILABLE\n"<<endl;
                return false;
                }
        }
}
bool cancelRoom(int rn){
        roomNode *newNode=new roomNode;
        newNode=findRoom(rn);
        if(!newNode){
                cout << "ROOM NUMBER "<<rn<<"CAN NOT BE FOUND\n"<<endl;
                return false;
        }
        else {
                if( newNode->status==1){
                        newNode->status=0;
                         cout << "ROOM NUMBER "<<rn<<"IS SUCCESSFULLY 
CANCELLED\n"<<endl;
                return true;
                }
                else{
                         cout << "BOOKING ROOM NUMBER "<<rn<<"CAN NOT BE 
FOUND\n"<<endl;
                         cout<< "UNABLE TO MAKE CANCELLATION"<<endl;
                return false;
                }
        }
}
// };


bool findBookRoom(int rn){
    roomNode*newNode=new roomNode;
    newNode=findRoom(rn);
    if(!newNode){
        return false;
    }
    else{
        if(newNode->status==1){
            return true;
        }
        else{
            return false;
        }
    }
}
roomNode* findRoom(int rn){
    roomNode *currNode=head;
    int currIndex=1;
    while(currNode&&currNode->roomNum!=rn){
        currIndex++;
    }
    if(currNode)
        return currNode;
    else
        return 0;
}
void displayRoom(){
    int num=0;
    roomNode*currNode=head;
    cout<<"===========ROOM IN HOTEL=========="<<endl;
    cout<<"----------------------------------"<<endl;
    cout<<"FLOOR\tROOM NO.\tROOM TYPE\tPRICE\tSTATUS"<<endl;

    while(currNode!=NULL){
        cout<<currNode->floorNo<<"\t"<<currNode->roomNum<<"\t\t"<<rt(currNode-
>roomType)<<"\t\t"<<currNode->price<<"\t"<<status(currNode->status)<<"\t"<<endl;
     currNode=currNode->next;
     num++;
    }
    cout<<"\nNUMBER OF ROOM IN HOTEL:"<<num<<endl<<endl<<endl;
 }
};
int getDays(string date) {
    int year=atoi(date.substr(0,4).c_str());
    int month= atoi(date.substr(4,2).c_str());
    int day= atoi(date.substr(6,2).c_str());
    int ans=0;
    for (int i=1900;i<year;++i){
        if (((year%4==0)&&(year%100!=0))|| (year%400==0)) {
            ans+=366;
        } else {
            ans+=365;
        }
    }
    for (int i=1; i<month;i++) {
        switch(i){
            case 1: ans+=31; break;
            case 2: ans+=(((year%4==0)&&(year%100!=0))||(year%400==0))?29:28;
break;
            case 3: ans+= 31; break;
            case 4: ans+= 30; break;
            case 5: ans+= 31; break;
            case 6: ans+= 30; break;
            case 7: ans+= 31; break;
            case 8: ans+= 31; break;
            case 9: ans+= 30; break;
            case 10:ans+= 31; break;
            case 11:ans+= 30; break;
            case 12:ans+= 31; break;
        }
    }
    return ans+=day-1;
}
int daysBetweenDates(string date1, string date2) {
    return abs(getDays(date1)-getDays(date2));
}

int main(){
        //CREATE HOTEL ROOM
    room h;
    customer cn;
    billQueue bq;
    int choice;
    //ADD DEFAULT ROOM
    int
    defaultroom[]={1101,1202,1303,1102,1103,1104,2201,2202,2203,3301,3302,3303,33
03,4401,4401};
    for (int x=0;x<15;x++)
    {
       h.insertSortedRoom(defaultroom[x]);
    }
    do{
        // system("CLS");
        cout<<"==========    HOTEL BOOKING SYSTEM     =========="<<endl;
        cout<<"\t\t\tLOGIN AS"<<endl;
        cout<<"\t\t\t1. ADMIN\n\t\t\t2. CUSTOMER \n\t\t\t3. EXIT\n"<<endl;
        cout<<"CHOICE : ";
        cin>>choice;
        switch(choice){
            case 1:{
                int ch1;
                int rn;
                int cont1;
                do{
                    system("CLS");
                    cout<<"========== ADMIN MENU =========="<<endl;
                    cout<<"\t1. ADD ROOM\n\t2. DELETE ROOM\n\t3. DISPLAY
ROOM\n\t4. CHANGE PRICE\n\t5. CHECK CUSTOMER BOOKING LIST\n\t6. REVIEW AND
CONFIRM PENDING BOOKING\n\t7. BACK TO MAIN MENU\n"<<endl;
                    cout<<"CHOICE : ";
                    cin>>ch1;
                    switch(ch1){
                        case 1:
                        cont1=1;
                            cout<<"\nENTER THE ROOM NUMBER YOU WANT TO ADD : ";
                            cin>>rn;
                        //     if(!h.findRoom(rn)){
                                h.insertSortedRoom(rn);
                                cout<<"YOUR SUCCESSFULLY ADD ROOM NUMBER("
<<rn<<") TO THE LIST"<<endl;
                        //        }
                        //        else{
                        //             cout<<"ROOM NUMBERED "<<rn<<" EXISTED.
ENTER OTHER ROOM NUMBER." <<endl;
                        //             cout<<"DO YOU WANT TO CONTINUE TO ADD MORE
ROOM (1-YES, 0-NO):";
                        //             cin>>cont1;
                        //         }
                        break;
                        case 2:
                        cont1=1;
                        do{
                            cout<<"\nENTER THE ROOM NUMBER YOU WANT TO DELETE :";
                            cin>>rn;
                            if(h.findRoom(rn)){
                                h.deleteRoom(rn);
                            }
                            else{
                                cout<<"ROOM CANNOT BE FOUND! ENTER OTHER ROOM
NUMBER."<<endl;
                            }
                            cout<<"DO YOU WANT TO CONTINUE TO DELETE MORE ROOM(1-
YES, 0-NO):";
                            cin>>cont1;
                         }while(cont1!=0);
```

```
                            break;
                            case 3:
                            h.displayRoom();
                            break;
                            case 4:
                            cont1=1;
                            do{
                                cout<<"\nENTER THE ROOM  NUMBER THAT YOU WANT
CHANGE PRICE: "<<endl;
                                cout<<"ROOM NUMBER        :";
                                cin>>rn;
                                if(h.findRoom(rn)){
                                    float p;
                                    cout<<"CHANGE TO PRICE RM :";
                                    cin>>p;
                                    roomNode *newNode=h.findRoom(rn);
                                    newNode->price=p;
                                    cout<<"YOU SUCCESSFULLY CHANGE THE PRICE OF
ROOM "<<rn<< "TO PRICE RM"<<p<<endl<<endl;
                                }
                                else{
                                    cout<<"ROOM NUMBER ("<<rn<<") CANNOT BE
FOUND. ENTER OTHER ROOM NUMBER."<<endl<<endl;
                                }
                                cout<<"DO YOU WANT TO CONTINUE TO CHANGE PRICE OF
OTHER ROOMS (1-YES,0-NO):";
                                cin>>cont1;
                               }while(cont1!=0);
                               break;
                               case 5:
                               cn.displayCustRoom();
                               break;
                               case 6:
                               cont1=1;
                               cout<<"\n\n======    START TO REVIEW BOOKING
INFORMATION ======\n";
                               do{
                                billNode* rno=new billNode;
                                rno=bq.getFrontRN();
                                string con=bq.dequeue();
                                roomNode *t=new roomNode;
                                t=h.findRoom(rno->roomNum);
                                if(con=="BOOKED & CONFIRMED"){
                                    t->status=0;
                                    break;
                                }else if(con=="BOOKED & PENDING"){
                                    t->status=1;
                                    bq.enqueue(rno->roomNum,rno->price,rno-
>ic,rno->name,rno->DateIn,rno->DateOut,rno->days);
                                    break;
                                }
                                else{}
                                cout<<"DO YOU WANT TO CONTINUE TO REVIEW(1-YES,0-
NO):";

                                cin>>cont1;
                               } while(cont1!=0);
                               break;
                               case 7:
                                      break;
                               default:
                               cout<<"INVALID CHOICE."<<endl;
                               break;
                               }
                               system("pause");
                              getch();
                }while(ch1!=7);
                break;
                }
                           //     cas1 end
                      break;

             case 2:{
            int ch2;
            int rn,rn2;
            int cont2;
                do{
                      system("CLS");
                      cout<<"=======COUSTOMER MENUE========"<<endl;
                      cout<<"\t1.DISPLAY ROOM\n\t2.BOOK A ROOM\n\t3.CANCEL
ROOM\n\t4.BACK TO MAIN MENUE"<<endl;
                      cout<<"CHOICE :";
                      cin>>ch2;
                      switch(ch2){
                      case 1:
                           h.displayRoom();
                           break;
                      case 2:
                           cont2=1;
                           do{
                           cout<<"\nENTER THE ROOM NUMBER YOU WANT TO BOOK:";
                           cin>>rn;
                           bool book=h.bookRoom(rn);
                           if(book){
                           string n,p,ic,in,out;
                           int din,min,yin,dout,mout,yout;
                           char s;
                           cout<<"=======YOUR PERSONAL DETAILS========"<<endl;
                           cout<<"ENTER YOUR NAME :";
                           cin.ignore();
                           getline(cin,n);
                           cout<<"ENTER YOUR PHONE NO :";
                           getline(cin,p);
                           cout<<"ENTER YOUR IC(xxxxxx-xx-xxxx):";
                           getline(cin,ic);
                           cout<<"DATE IN (YYYYMMDD) :";
                           getline(cin,out);
                           int days=daysBetweenDates(in,out);
                           cn.sortedCustIn(n,p,ic,in,out,rn);
                           cout<<"\n-----------------------\n";
                           cout<<"\nPLEASE PROCEED TO BILL AT COUNTER\n";
                           cout<<"\n-----------------------\n";
                           roomNode*t=new roomNode;
                           t=h.findRoom(rn);
                           bq.enqueue(rn,t->price,ic,n,in,out,days);
                                }
                            cout<<"DO YOU WANT TO CONTINUE TO BOOK ROOM(1-YES,0-
N0):";

                              cin>>cont2;
                        } while(cont2!=0);
                          break;

                          case 3:
                          cont2=1;
                               do{
                                    string ic2;
                                    int rn2=0;
                                    cout<<"\nENTER THE ROOM NUMBER YOU WANT TO
CANCEL :";

                                    cin>>rn2;
                                    cout<<"ENTER YOUR IC TO MAKE
CANCELLATION(xxxxxx-xx-xxxx):";

                                    cin.ignore();
                                    getline(cin,ic2);
                                    bool targetRoom=h.findBookRoom(rn2);
                                    if(targetRoom){
```

```
                                    bool targetCust=cn.findCust(ic2);
                                    if(targetCust){
                                    h.cancelRoom(rn2);
                                    cn.deleteCust(ic2,rn2);
                                    }
                                    else{
                                         cout<<"\nRECORD CANNOT BE FOUND\nUNABLE
TO MAKE CANCELLATION"<<endl;
                                        }
                                                }
                                     else{
                                         cout<<"RECORD CANNOT BE FOUND\nUNABLE TO
MAKE CANCELLATION"<<endl;
                                        }
                                        cout<<"DO YOU WANT TO CONTINUE TO CANCEL
ROOM (1-YES,0-NO):";

                                        cin>>cont2;
                                } while(cont2!=0);
                                break;
                                case 4:
                                  break;
                                  default:
                                  cout<<"INVALID CHOICE."<<endl;
                                  break;
                      }
                         system("pause");
                         getch();
                }while(ch2!=4);
                        break;
// cas 2 end
                case 3:
                cout<<"THANK YOU! BYE BYE."<<endl;
               exit(0);
               default:
               cout<<"INVALID CHOICE."<<endl;
               break;
}
//  switch ends       //
 getch();
}
return 0;
}while(choice!=3);
}
```

Conclusion:

At this moment on the world everything is going computerized from small things to huge things which has brought a huge change on our lives. A few years ago computer was not even considered as an essential but just a rich person's toy but the time has changed and as we all know it has a huge impact on our lives. Huge companies like Apple, Microsoft and many others companies all are investing to make a better future with efficiency. (Wanghi, 2018) Tourism has boosted all around the world and hotels need to manage their data properly and taxes well. Majorparts of the world need to manage their customers as anything can happen and even data which were over a decade ago can have a huge impact on some cases. People often makes some mistakes and with a proper management system those error will be deleted. Data will be stored properly and bills with some features with of the hotel can be added easily without any errors. Sometime people forget the things needed by the customer but with proper management then it can't be forgotten and easy management

References:

- https://www.studocu.com/in/document/international-university-vnu-hcm/internship/dsa-project-report-fdfdsfg/60339374
- https://www.studocu.com/en-gb/document/coventry-university/product-design-bsc-final-project-work/final-year-assignment-hotel-management-system/15490398
- https://en.wikipedia.org/wiki/Hotel_Management