

**CHAROTAR UNIVERSITY OF SCIENCE AND
TECHNOLOGY**

**DEVANG PATEL INSTITUTE OF ADVANCE
TECHNOLOGY AND RESEARCH**

NAME: PARTH N PATEL

SUBJECT: DATA STRUCTURE AND ALGORITHM

CODE: CE 245

SEMESTER: 4

PRACTICAL-1

NO	CONTENT
1	<p><u>AIM:</u></p> <p>Write a program to store roll numbers of student in array who attended training program in random order. Write functions for Search whether particular student attended training program or not using various searching algorithms.</p> <p><u>PROGRAM CODE:</u></p> <pre>#include<iostream>> using namespace std; short linearSearch(int arr[],short len,short key) { for(short i=0;i<len;i++) { if(arr[i]==key) return i; } return -1; } short binarySearch(int arr[],short len,short key) { short lb=0,ub=len,mid=0; while(lb<=ub) { mid=(lb+ub)/2; if(arr[mid]==key)</pre>

```
        return mid;
    else if(arr[mid]<key)
    {
        lb=mid+1;
    }
    else
    {
        ub=mid-1;
    }
}
return -1;
}
int main()
{
    int arr[10]={ 1,4,9,27,29,98,102,133,135,151 };
    short key;
    cout<<"Enter the roll number for linear search: ";
    cin>>key;
    short ans1=linearSearch(arr,10,key);

    cout<<"Enter the roll number for binary search: ";
    cin>>key;
    short ans2=binarySearch(arr,10,key);

    cout<<"Result of linear Search: "<<endl;
    if(ans1== -1)
    {
        cout<<"This Roll number has not attended the training program!!"<<endl;
    }
    else
    {
        cout<<"Roll Number : "<<arr[ans1]<<" has attended the training
program!!"<<endl;
```

```
}

cout<<endl;
cout<<"Result of Binary Search:"<<endl;
if(ans2==-1)
{
    cout<<"This Roll number has not attended the training program!!"<<endl;
}
else
{
    cout<<"Roll Number : "<<arr[ans2]<<" has attended the training
program!!"<<endl;
}

cout<<"PARTH PATEL"<<endl<<"19DCS098"<<endl;

return 0;

}
```

OUTPUT:

```
Enter the roll number for linear search: 98
Enter the roll number for binary search: 205
Result of linear Search:
Roll Number : 98 has attended the training program!!

Result of Binary Search:
This Roll number has not attended the training program!!
PARTH PATEL
19DCS098
```

CONCLUSION:

In this practical, we learned various searching algorithms.

PRACTICAL-2

N O	CONTENT
2	<p><u>AIM:</u></p> <p>Mark purchased Books from books store of standard 1 to 7. He purchased 4 books for each standard (for std.1 books are 1.1,1.2,1.3,1.4 and for std. 2 books are 2.1,2.2,2.3,2.4 and so on...). When he reached home, he opens the bag and sees that all the books got mixed. So, how he will sort all the books, according to the standards and their preference in that particular standard. (ex.: preference in std. 1 is 1.1<1.2)</p> <p><u>PROGRAM CODE:</u></p> <pre> #include<iostream> using namespace std; void display(float arr[],int len) { for(int i=0;i<len;i++) cout<<arr[i]<<" "; cout<<endl; } void Swap(float *x,float *y) { float tmp; tmp=*x; *x=*y; *y=tmp; </pre>

```
}

void bubbleSort(float arr[],int len,short order)
{
    if(order==1){
        for(int i=0;i<len-1;i++)
        {
            for(int j=0;j<len-1-i;j++)
            {

                if(arr[j]>arr[j+1])
                {
                    Swap(&arr[j],&arr[j+1]);
                }
            }
        }
    }
    else if(order==0)
    {
        for(int i=0;i<len-1;i++)
        {
            for(int j=0;j<len-1-i;j++)
            {

                if(arr[j]<arr[j+1])
                {
                    Swap(&arr[j],&arr[j+1]);
                }
            }
        }
    }
}
```

```
}

void insertionSort(float arr[], int len,short order)
{
    int i,j;
    float key;
    if(order==1){
        for (i = 1; i < len; i++)
        {
            key = arr[i];
            j = i - 1;

            while (j >= 0 && arr[j] > key)
            {
                arr[j + 1] = arr[j];
                j = j - 1;
            }
            arr[j + 1] = key;
        }
    }

    else if(order==0)
    {
        for (i = 1; i < len; i++)
        {
            key = arr[i];
            j = i - 1;

            while (j >= 0 && arr[j]<key)
            {
                arr[j + 1] = arr[j];
```

```
        j = j - 1;
    }
    arr[j + 1] = key;
}
}
}

int main()
{

    float
arr[]={ 7.4,1.4,2.4,3.4,6.4,4.4,5.4,7.3,5.3,6.3,2.3,1.3,3.3,4.3,6.2,7.2,5.2,4.2,1.2,2.2,3.2,
7.1,5.1,6.1,2.1,1.1,3.1,4.1};
    int len=sizeof(arr)/sizeof(arr[0]);
    cout<<"Insertion Sort in Ascending Order : "<<endl;
    insertionSort(arr,len,1);
    display(arr,len);
    cout<<"-----
-----"<<endl;

    cout<<"Insertion Sort in Descending Order : "<<endl;
    insertionSort(arr,len,0);
    display(arr,len);
    cout<<"-----
-----"<<endl;

    cout<<"Bubble Sort in Ascending Order : "<<endl;
    bubbleSort(arr,len,1);
    display(arr,len);
    cout<<"-----
-----"<<endl;

    cout<<"Bubble Sort in Descending Order : "<<endl;
    bubbleSort(arr,len,0);
    display(arr,len);
```



```

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

cout<<"PARTH PATEL\n19DCS098"<<endl;
return 0;

}

```

OUTPUT:

```

Insertion Sort in Ascending Order :
1.1 1.2 1.3 1.4 2.1 2.2 2.3 2.4 3.1 3.2 3.3 3.4 4.1 4.2 4.3 4.4 5.1 5.2 5.3 5.4 6.1 6.2 6.3 6.4 7.1 7.2 7.3 7.4
-----
Insertion Sort in Descending Order :
7.4 7.3 7.2 7.1 6.4 6.3 6.2 6.1 5.4 5.3 5.2 5.1 4.4 4.3 4.2 4.1 3.4 3.3 3.2 3.1 2.4 2.3 2.2 2.1 1.4 1.3 1.2 1.1
-----
Bubble Sort in Ascending Order :
1.1 1.2 1.3 1.4 2.1 2.2 2.3 2.4 3.1 3.2 3.3 3.4 4.1 4.2 4.3 4.4 5.1 5.2 5.3 5.4 6.1 6.2 6.3 6.4 7.1 7.2 7.3 7.4
-----
Bubble Sort in Descending Order :
7.4 7.3 7.2 7.1 6.4 6.3 6.2 6.1 5.4 5.3 5.2 5.1 4.4 4.3 4.2 4.1 3.4 3.3 3.2 3.1 2.4 2.3 2.2 2.1 1.4 1.3 1.2 1.1
-----
PARTH PATEL
19DCS098

```

CONCLUSION:

In this practical, we learned various sorting algorithm

PRACTICAL-3

NO	CONTENT
3	<p><u>AIM:</u></p> <p>Sometimes a program requires two stacks containing the same type of items. If the two stacks are stored in separate arrays. Then one stack might overflow while there was considerable unused space in the other. A neat way to avoid the problem is to put all the space in one array and let one stack grow from one end of the array and the other stack start at the other end and grow in opposite direction i.e., toward the first stack, in this way, if one stack turns out to be large and the other small, then they will still both fit, and there will be no overflow until all the space is actually used. Declare a new structure type Double stack that includes the array and the two indices top A and top B, and write functions Push A, Push B, Pop A and Pop B to handle the two stacks with in one Double Stack.</p> <p><u>PROGRAM CODE:</u></p> <pre>#include<iostream> using namespace std; class DoubleStack { int* arr; int size; int topA,topB; public: void initialize() { cout<<"Enter the size of Stack : ";</pre>

```
cin>>size;
arr=new int[size];
topA=-1;
topB=size;
}

void pushA(int x)
{
    if(topA<topB-1)
    {
        topA++;
        arr[topA]=x;
        cout<<"Element "<<x<<" added to stack!!"<<endl;
    }
    else
    {
        cout<<"Stack Overflow !!!\n Please try to delete some elements!!"<<endl;
    }
}

void pushB(int x)
{
    if(topA<topB-1)
    {
        topB--;
        arr[topB]=x;
        cout<<"Element "<<x<<" added to stack!!"<<endl;
    }
    else
    {
        cout<<"Stack Overflow !!!\n Please try to delete some elements!!"<<endl;
    }
}
```

```
int popA()
{
    if(topA>=0)
    {
        int tmp=arr[topA];
        topA--;
        return tmp;
    }
    else
    {
        return -1;
    }
}

int popB()
{
    if(topB<size)
    {
        int tmp=arr[topB];
        topB++;
        return tmp;
    }
    else
    {
        return -1;
    }
}
```

```
int returnSize()
{
    return size;
}

};

int main()
{
    DoubleStack s;

    s.initialize();
    s.pushA(1);
    s.pushA(2);
    s.pushA(3);
    s.pushA(4);
    s.pushA(5);
    s.pushB(10);
    s.pushB(9);
    s.pushB(8);
    s.pushB(7);
    s.pushB(6);

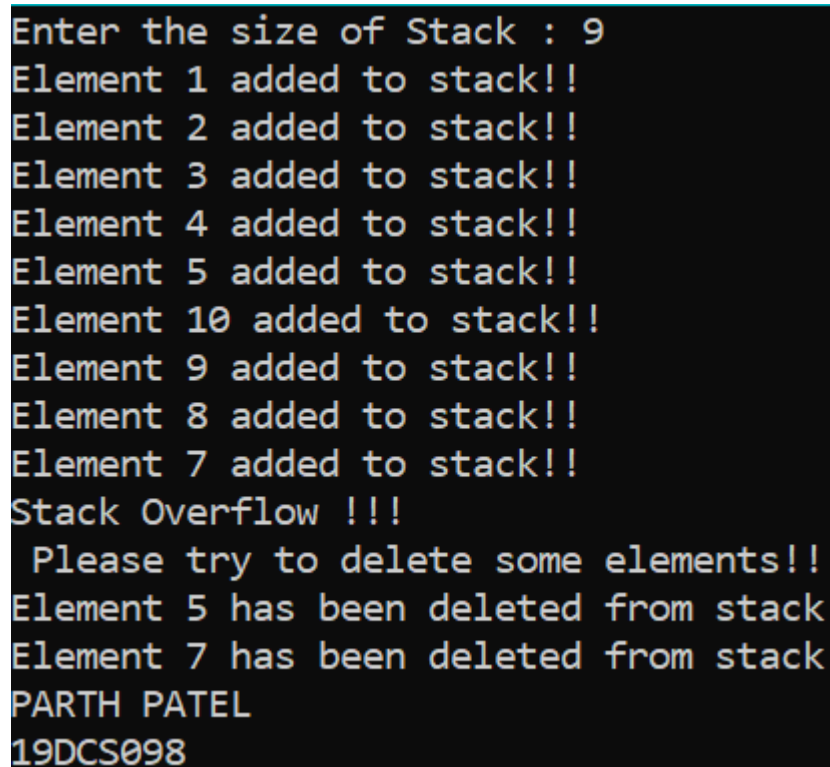
    int x=s.popA();
    if(x!=-1)
        cout<<"Element "<<x<<" has been deleted from stack"<<endl;
    else
        cout<<"Stack Underflow!!!\n Seems stack is empty"<<endl;

    x=s.popB();
    if(x!=-1)
        cout<<"Element "<<x<<" has been deleted from stack"<<endl;
    else
```

```
cout<<"Stack Underflow!!!\n Seems stack is empty"<<endl;

cout<<"PARTH PATEL\n19DCS098"<<endl;

return 0;
}
```

OUTPUT:

```
Enter the size of Stack : 9
Element 1 added to stack!!
Element 2 added to stack!!
Element 3 added to stack!!
Element 4 added to stack!!
Element 5 added to stack!!
Element 10 added to stack!!
Element 9 added to stack!!
Element 8 added to stack!!
Element 7 added to stack!!
Stack Overflow !!!
Please try to delete some elements!!
Element 5 has been deleted from stack
Element 7 has been deleted from stack
PARTH PATEL
19DCS098
```

CONCLUSION:

In this practical, we learned the concept of stack.

PRACTICAL-4

NO	CONTENT
4	<p><u>AIM:</u></p> <p>Write a program to reverse a string using appropriate data structure. (Do not use any predefine function)</p> <p><u>PROGRAM CODE:</u></p> <pre>#include<iostream> using namespace std; class Stack { char* arr; int size; int top; public: void initialize() { top=-1; cout<<"Enter the size of string : "; cin>>size; arr= new char[size]; } }</pre>

```
void push(char x)
{
    if(top<size-1)
    {
        top++;
        arr[top]=x;
    }
    else
        cout<<"Stack Overflow!!!"<<endl;

}

char pop()
{
    if(top== -1)
        {cout<<"UnderFlow!!!"<<endl;
        return '0';}
    else
    {
        char x=arr[top];
        top--;
        return x;
    }
}

int returnSize()
{
    return size;
}
```



```
    }

};

int main()
{
    Stack s;
    s.initialize();

    char c[s.returnSize()];
    cout<<"Enter the string : ";
    cin>>c;

    for(int i=0;i<s.returnSize();i++)
        s.push(c[i]);

    for(int i=0;i<s.returnSize();i++)
        c[i]=s.pop();

    cout<<"The reversed String is : "<<c<<endl;

    cout<<"PARTH PATEL\n19DCS098"<<endl;
    return 0;
}
```

OUTPUT:

```
Enter the size of string : 7
Enter the string : DEPSTAR
The reversed String is : RATSPED
PARTH PATEL
19DCS098
```

CONCLUSION:

In this practical, we learned to reverse string using stack

PRACTICAL-5

NO	CONTENT
5	<p><u>AIM:</u></p> <p>We are developing software for a call center. When a client calls, his/her call should be stored until there is a free service representative to pick the call. Calls should be processed in the same order they are received. Select appropriate data structure and build call center software system.</p> <p><u>PROGRAM CODE:</u></p> <pre>#include<iostream> using namespace std; class Queue { int size; int front,rear; int *arr; public: void initialize() { cout<<"Enter the size of Queue : "; cin>>size; arr= new int[size]; front=0; rear=-1; } }</pre>

```
bool isFull()
{
    if(rear==size-1)
        return true;
    else
        return false;
}
```

```
bool isEmpty()
{
    if(front > rear)
        return true;
    else
        return false;
}
```

```
int peek()
{
    return arr[front];
}
```

```
void enqueue(int x)
{
    if(isFull())
    {
        cout<<"Sorry !!! The phone line is full.\nTry again later"<<endl;
    }
    else if(isEmpty())
    {
        rear++;
        arr[rear]=x;
        cout<<"Welcome to CHARUSAT!!\nYour call has been placed in Queue.\n
Your Token Number : "<<rear+1<<endl;
        cout<<"-----"<<endl;
    }
    else
    {
        rear++;
        arr[rear]=x;
        cout<<"Welcome to CHARUSAT!!\nYour call has been placed in Queue.\n
Your Token Number : "<<rear+1<<endl;
        cout<<"Please Wait till our call representative pick your call!!"<<endl;
        cout<<"-----"<<endl;
        cout<<"Currently Token number : "<<front<<" is in call"<<endl;
        cout<<"-----"<<endl;
    }
}

void dequeue()
{
```

```
        if(isEmpty())
        {
            cout<<"There are no customers in line."<<endl;
            cout<<"-----"<<endl;
        }

        else
        {
            cout<<"You have been disconnected from the call."<<endl;
            cout<<"Your Token Number was : "<<front+1<<endl;
            cout<<"Thank you !!"<<endl;
            cout<<"-----"<<endl;
            front++;
        }
    }

};

int main()
{
    Queue q;
    q.initialize();
    for(int i=1;i<5;i++)
        q.enqueue(i);
    for(int i=1;i<=3;i++)
        q.dequeue();

    cout<<endl;
    cout<<"PARTH PATEL\n19DCS098"<<endl;
    return 0;
}
```

OUTPUT:

```
Enter the size of Queue : 5
Welcome to CHARUSAT!!
Your call has been placed in Queue.
  Your Token Number : 1
-----
Welcome to CHARUSAT!!
Your call has been placed in Queue.
  Your Token Number : 2
Please Wait till our call representative pick your call!!
-----
Currently Token number : 0 is in call
-----
Welcome to CHARUSAT!!
Your call has been placed in Queue.
  Your Token Number : 3
Please Wait till our call representative pick your call!!
-----
Currently Token number : 0 is in call
-----
Welcome to CHARUSAT!!
Your call has been placed in Queue.
  Your Token Number : 4
Please Wait till our call representative pick your call!!
-----
Currently Token number : 0 is in call
-----
You have been disconnected from the call.
Your Token Number was : 1
Thank you !!
-----
```

```
You have been disconnected from the call.  
Your Token Number was : 2  
Thank you !!  
-----
```

```
You have been disconnected from the call.  
Your Token Number was : 3  
Thank you !!  
-----
```

```
PARTH PATEL  
19DCS098
```

CONCLUSION:

In this practical, we learned the concept of Queue

PRACTICAL-6

NO	CONTENT
6	<p><u>AIM:</u></p> <p>Write a program to implement Circular Queue with all operations. Check the queue contents and conditions with different combinations of insert and delete operations. Show the content of circular queue with front and rear pointer after each operation. Initially, the queue is empty. The size of the queue is 5. The sequence of operation given below: 2 • Insert 10, 50, 40, 80 • Delete • Insert 200, 70, 150 • Delete • Delete • Delete</p> <p><u>PROGRAM CODE:</u></p> <pre>#include<iostream> using namespace std; class CircularQueue { int rear,front,size; int* arr; public: void initialize() { cout<<"Enter the size : "; cin>>size; arr=new int[size]; front=rear=-1; } }</pre>

```
void display()
{

    cout<<"-----"<<endl;
    if(front==-1)
        cout<<"Circular Queue is empty!!"<<endl;
    else if(rear>=front)
    {
        for(int i=front;i<=rear;i++)
            cout<<arr[i]<<" ";
        cout<<endl;

    }
    else{
        for (int i = front; i < size; i++)
            cout<<arr[i]<<" ";
        for (int i = 0; i <= rear; i++)
            cout<<arr[i]<<" ";
        cout<<endl;
    }
}

void enqueue(int x)
{
    if((front==0 &&rear==size-1)|| rear==front-1)
        cout<<"Overflow!!"<<endl;
    else if(front==-1)
    {
        front=rear=0;
        arr[rear]=x;
```

```
    }

    else if(rear==size-1 && front!=0)
    {
        rear=0;
        arr[rear]=x;
    }
    else
    {
        rear++;
        arr[rear]=x;
    }

    display();
}

void dequeue()
{
    if(front==-1)
        cout<<"Circular Queue is empty!!!"<<endl;

    if(front==rear)
    {
        front=rear=-1;
    }
    else if(front==size-1)
    {
        front=0;
    }
    else
        front++;
}
```

```
        display();

    }
};

int main()
{
    CircularQueue cq;
    cq.initialize();
    cq.enqueue(10);
    cq.enqueue(50);
    cq.enqueue(40);
    cq.enqueue(80);
    cq.dequeue();
    cq.enqueue(200);
    cq.enqueue(70);
    cq.enqueue(150);
    cq.dequeue();
    cq.dequeue();
    cq.dequeue();

    cout<<"PARTH PATEL\n19DCS098"<<endl;
    return 0;
}
```

OUTPUT:

```
Enter the size : 5
-----
10
-----
10 50
-----
10 50 40
-----
10 50 40 80
-----
50 40 80
-----
50 40 80 200
-----
50 40 80 200 70
Overflow!!
-----
50 40 80 200 70
-----
40 80 200 70
-----
80 200 70
-----
200 70
PARTH PATEL
19DCS098
```

CONCLUSION:

In this practical, we learned the concept of circular queue

PRACTICAL-7

NO	CONTENT
7	<p><u>AIM:</u></p> <p>Write a Program to implement Double Ended Queue operations. (insert, delete, display)</p> <p><u>PROGRAM CODE:</u></p> <pre>#include<iostream> using namespace std; class deque { int size,front,rear; int* arr; public: void initialize() { cout<<"Enter the size : "; cin>>size; arr=new int[size]; front=rear=-1; } }</pre>

```
void enqueue_front(int x)
{
    if((front==0 && rear==size-1)|| (front==rear+1))
        cout<<"Overflow"<<endl;
    else if(front== -1 && rear== -1)
    {
        front=rear=0;
        arr[front]=x;
    }
    else if(front==0)
    {
        front=size-1;
        arr[front]=x;
    }
    else
    {
        front=front-1;
        arr[front]=x;
    }
}

void enqueue_rear(int x)
{
    if((front==0 && rear==size-1)&&(front==rear+1))
        cout<<"Overflow!!"<<endl;
    else if(front== -1 && rear== -1)
    {
        rear=0;
        arr[rear]=x;
    }
}
```

```
        else if(rear==size-1)
        {
            rear=0;
            arr[rear]=x;
        }
        else
        {
            rear++;
            arr[rear]=x;
        }
    }

void dequeue_front()
{
    if(front==1 && rear==0)
        cout<<"Underflow!!"<<endl;
    else if(front==rear)
    {
        front=rear=-1;
    }
    else if(front==size-1)
    {
        front=0;
    }
    else
    {
        front++;
    }
}
```



```
void dequeue_rear()
{
    if(front==-1 && rear==-1)
        cout<<"Underflow!!"<<endl;
    else if(front==rear)
        front=rear=-1;
    else if(rear==0)
    {
        rear=size-1;
    }
    else
    {
        rear=rear-1;
    }
}
```

```
void display()
{
    int i=front;
    cout<<"Elements : ";
    while(i!=rear)
    {
        cout<<arr[i]<<" ";
        i=(i+1)%size;
    }
    cout<<endl;
}
```

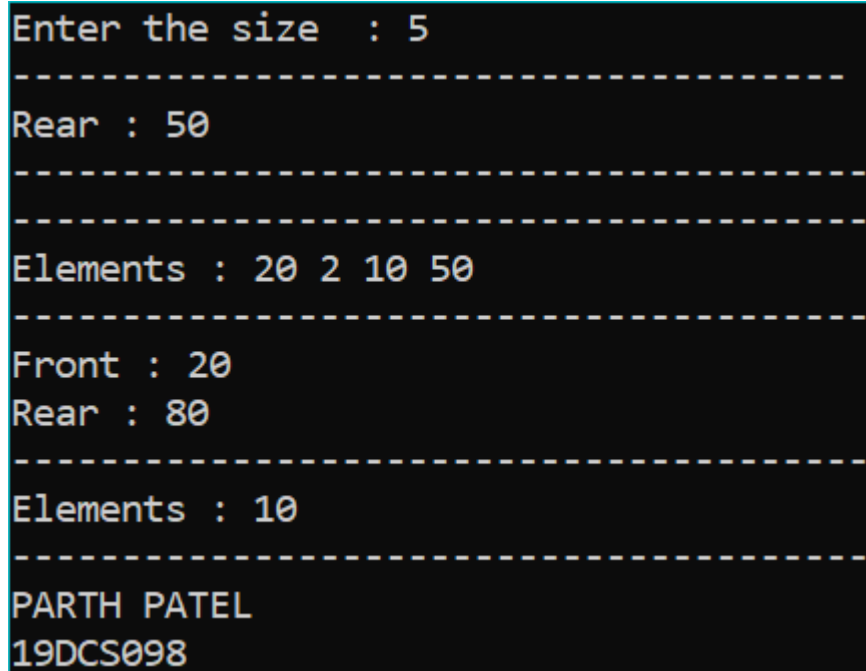
```
void getFront()
{
    if(front==-1 && rear==-1)
        cout<<"Queue is Empty!!"<<endl;
    else
        cout<<"Front : "<<arr[front]<<endl;
}

void getRear()
{
    if(front==-1 && rear==-1)
        cout<<"Queue is Empty!!"<<endl;
    else
        cout<<"Rear : "<<arr[rear]<<endl;
}
};

int main()
{
    deque d;
    d.initialize();
    d.enqueue_front(10);
    d.enqueue_rear(50);
    cout<<"-----"<<endl;
    d.getRear();
    cout<<"-----"<<endl;
    d.enqueue_front(2);
    d.enqueue_front(20);
    d.enqueue_rear(80);
    cout<<"-----"<<endl;
    d.display();
    cout<<"-----"<<endl;
    d.getFront();
```

```
d.getRear();  
d.dequeue_front();  
d.dequeue_rear();  
d.dequeue_front();  
cout<<"-----"<<endl;  
d.display();  
cout<<"-----"<<endl;  
  
cout<<"PARTH PATEL\n19DCS098"<<endl;  
return 0;  
}
```

OUTPUT:



```
Enter the size : 5  
-----  
Rear : 50  
-----  
Elements : 20 2 10 50  
-----  
Front : 20  
Rear : 80  
-----  
Elements : 10  
-----  
PARTH PATEL  
19DCS098
```

CONCLUSION:

In this practical, we learned the concept of double ended queue

PRACTICAL-8

NO	CONTENT
8	<p><u>AIM:</u></p> <p>DEPSTAR has student's club named 'Pinnacle Club'. Students of Second, third and final year of department can be granted membership on request. Similarly, one may cancel the membership of club. First node is reserved for president of club and last node is reserved for secretary of club. Write a program to maintain club member 's information using appropriate linked list. Store student PRN and Name. Write functions to a) Add and delete the members as well as president or even secretary. b) Compute total number of members of club c) Display members d) Display list in reverse order using recursion e) Two linked lists exist for two divisions. Concatenate two lists.</p> <p><u>PROGRAM CODE:</u></p> <pre> #include <iostream> #include<string> using namespace std; class list; class node { int prn; string name; node *next; public: node(int pr,string tmp_name) { </pre>

```
        prn=pr;
        next=NULL;
        name=tmp_name;
    }

friend class list;
};
class list
{
    node *start;
public:
    list(){
        start=NULL;
    }
    bool reverseDisplay()
    {
        if(start==NULL)
            return false;
        node *temp=start;
        displayRev(temp);
        return true;
    }
    void displayRev(node *t)
    {
        if(t==NULL)
            return;
        else
        {
            displayRev(t->next);
            cout<<"PRN NO:"<<t->prn<<" Name: "<<t->name<<endl;
        }
    }
}
```

```
void create()
{
    int no;
    string name;
    if(start==NULL)
    {
        cout<<"Enter PRN number: ";
        cin>>no;
        cout<<"Enter name: ";
        cin>>name;
        start=new node(no,name);
        cout<<"LIST CREATED SUCCESSFULLY"<<endl;
    }
    else
    {
        cout<<"List Exists. Cannot create new"<<endl;
    }
}

void display()
{
    node *t;
    t=start;
    if(start==NULL)
        cout<<"List Empty";
    else
    { cout<<"LIST : "<<endl;
        while(t!=NULL){
            cout<<t->prn<<" "<<t->name<<" \n";
```

```
                                t=t->next;
                                }
                                }
                                }
void insertAtBeginning()
{
    int no;
    string nam;
    node *temp;
    if(start==NULL)
    {
        create();
    }
    else
    {
        cout<<"Enter PRN number: ";
        cin>>no;
        cout<<"Enter name: ";
        cin>>nam;
        temp=new node(no,nam);
        temp->next=start;
        start=temp;;
        cout<<"Inserted "<<temp->name<<" at root node.";
    }
}
void insertAtEnd()
{
    int no;
    string nam;
    node *t;
    if(start==NULL)
        create();
    else
```

```
{
    cout<<"Enter PRN number: ";
    cin>>no;
    cout<<"Enter name: ";
    cin>>nam;
    t=start;
    while(t->next!=NULL)
        t=t->next;

    node*p=new node(no,nam);
    t->next=p;
}
}
void insertAfter()
{
    int prev_no;
    cout<<"Enter PRN No:";
    cin>>prev_no;
    node *t;
    t=start;
    string nam;
    int flag=0,no;
    while(t!=NULL)
    {
        if(t->prn==prev_no)
        {
            flag=1;
            break;
        }
        t=t->next;
    }
    if(flag==1)
    {
```



```
        node *p;
        cout<<"Enter PRN number: ";
        cin>>no;
        cout<<"Enter name: ";
        cin>>nam;
        p=new node(no,nam);
        p->next=t->next;
        t->next=p;
    }
    else
    {
        cout<<prev_no<<" not in the list";
    }
}

void deleteAtFirst()
{
    node *t;
    if(start==NULL)
        cout<<"No members in club";
    else
    {
        t=start;
        start=start->next;
        t->next=NULL;
        delete t;
        cout<<"President removed/resigned from club";
    }
}
```

```
void deleteByValue()
{
    int no,flag=0;
    node *t,*prev;
    if(start==NULL)
        cout<<"No members in club";
    else
    {
        cout<<"Enter PRN number of member to be removed: ";
        cin>>no;
        t=start->next;
        while(t->next!=NULL)
        {
            if(t->prn==no){
                flag=1;
                break;
            }
            prev=t;
            t=t->next;
        }
        if(flag==1)
        {
            prev->next=t->next;
            t->next=NULL;
            delete t;
            cout<<"Member with prn no: "<<no<<" is deleted"<<endl;
        }
        else
            cout<<"Member not in the list."<<endl;
```

```
    }  
}  
  
void deleteAtEnd()  
{  
    node *t,*prev;  
    t=start;  
    if(start==NULL)  
        cout<<"No members in the club"<<endl;  
    else  
    {  
        while(t->next!=NULL)  
        {  
            prev=t;  
            t=t->next;  
        }  
        prev->next=NULL;  
        delete t;  
        cout<<"Secretary removed/resigned from the club"<<endl;  
    }  
}  
  
int computeTotal()  
{  
    node *t;  
    int count=0;  
    t=start;  
    if(start==NULL)  
    {  
        cout<<"No members in the club"<<endl;  
        return 0;  
    }  
    while(t!=NULL)
```

```
{
count++;
t=t->next;
}

return count;
}

void sortList()
{
    node *i,*j,*last=NULL;
    int tprn;
    string tname;

    if(start==NULL)
    {
        cout<<"No members in the club"<<endl;
        return ;
    }
    for(i=start;i->next!=NULL;i=i->next)
    {
        for(j=start;j->next!=last;j=j->next)
        {
            if((j->prn)>(j->next->prn))
            {
                tprn=j->prn;
                tname=j->name;
                j->prn=j->next->prn;
                j->name=j->next->name;

                j->next->prn=tprn;
                j->next->name=tname;
            }
        }
    }
}
```

```
        }
    }
}
cout<<"List is sorted"<<endl;
display();
}
void concatList(list &q1)
{
    node *t,*p;
    t=q1.start;
    if(t==NULL)
    {
        cout<<"List 2 empty";
        return;
    }
    p=start;
    while(p->next!=NULL)
    {
        p=p->next;
    }
    p->next=t;
    q1.start=NULL; //second list is set to null
    cout<<"After concatenation:"<<endl;
    display();
}

};

int main() {
    list *l;
    int choice,selectList;
    list l1,l2;
    l=&l1;
    start:
```

```

        cout<<"Select List:"<<endl<<"1. List 1"<<endl<<"2. List
2"<<endl<<"Enter your choice: ";
        cin>>selectList;

        if(selectList==1)
        {
            l=&l1;
        }
        else if(selectList==2)
        {
            l=&l2;
        }
        else
        {
            cout<<"Please select correct option"<<endl;
            goto start;
        }
        do
        {
            cout<<"\n1. create\n2.Insert President\n3.Insert secretary\n4.insert
after position(member)\n5.Display list"
            <<"\n6.Delete President\n7.Delete Secretary\n8.Delete
Member\n9.Find total No. of members\n10.Sort list\n11. Previous Menu1"
            <<"\n12.Combine lists\n13.Reverse Display\n0. Exit"<<endl;
            cout<<"Please Enter your choice : ";
            cin>>choice;

            switch(choice)
            {
                case 1: l->create();
                    break;
                case 2: l->insertAtBeginning();
                    break;
            }
        } while(choice!=0);
    }
}

```

```
case 3: l->insertAtEnd();
        break;
case 4: l->insertAfter();
        break;
case 5: l->display();
        break;
case 6: l->deleteAtFirst();
        break;
case 7: l->deleteAtEnd();
        break;
case 8: l->deleteByValue();
        break;
case 9: cout<<"\nTotal members: "<<l->computeTotal();
        break;
case 10: l->sortList();
        break;
case 11:
        goto start;
        break;
case 12:
        l1.concatList(l2);
        break;
case 13:
        l->reverseDisplay();
        break;
default:
        cout<<"Please select correct options"<<endl;
    }
    }while(choice!=0);
cout<<"PARTH PATEL\n19DCS098"<<endl;
return 0;
}
```

OUTPUT:

```
Select List:
1. List 1
2. List 2
Enter your choice: 1

1. create
2.Insert President
3.Insert secretary
4.insert after position(member)
5.Display list
6.Delete President
7.Delete Secretary
8.Delete Member
9.Find total No. of members
10.Sort list
11. Previous Menu1
12.Combine lists
13.Reverse Display
0. Exit
Please Enter your choice :
```

```
Please Enter your choice : 1
Enter PRN number: 1
Enter name: List-1
LIST CREATED SUCCESSFULLY
```



```
Please Enter your choice : 2
Enter PRN number: 98
Enter name: Parth
Inserted Parth at root node.
```

```
Please Enter your choice : 3
Enter PRN number: 133
Enter name: Narendra
```

```
Please Enter your choice : 9

Total members: 4
1 create
```

```
Please Enter your choice : 6
President removed/resigned from club
```

```
Please Enter your choice : 7
Secretary removed/resigned from the club
1 create
```

```
Please Enter your choice : 8
Enter PRN number of member to be removed: 2
Member not in the list.
```

CONCLUSION:

In this practical, we learned the concept of linked list

PRACTICAL-10

N O	CONTENT
10	<p><u>AIM:</u></p> <p>Write a menu driven program to construct Binary Search Tree of your mobile phone number and apply different tree traversal techniques on BST</p> <p><u>PROGRAM CODE:</u></p> <pre> #include<iostream> using namespace std; struct treeNode { int data; treeNode *left; treeNode *right; }; treeNode* FindMin(treeNode *node) { if(node==NULL) { /* There is no element in the tree */ return NULL; } if(node->left) /* Go to the left sub tree to find the min element */ return FindMin(node->left); else </pre>

```
        return node;
    }

    treeNode* FindMax(treeNode *node)
    {
        if(node==NULL)
        {
            /* There is no element in the tree */
            return NULL;
        }
        if(node->right) /* Go to the left sub tree to find the min element */
            return(FindMax(node->right));
        else
            return node;
    }

    treeNode *Insert(treeNode *node,int data)
    {
        if(node==NULL)
        {
            treeNode *temp;
            temp=new treeNode;
            //temp = (treeNode *)malloc(sizeof(treeNode));
            temp -> data = data;
            temp -> left = temp -> right = NULL;
            return temp;
        }
        if(data >(node->data))
        {
            node->right = Insert(node->right,data);
        }
        else if(data < (node->data))
```

```
{
    node->left = Insert(node->left,data);
}

/* Else there is nothing to do as the data is already in the tree. */
return node;
}

treeNode * Find(treeNode *node, int data)
{
    if(node==NULL)
    {
        /* Element is not found */
        return NULL;
    }
    if(data > node->data)
    {
        /* Search in the right sub tree. */
        return Find(node->right,data);
    }
    else if(data < node->data)
    {
        /* Search in the left sub tree. */
        return Find(node->left,data);
    }
    else
    {
        /* Element Found */
        return node;
    }
}
```

```
void Inorder(treeNode *node)
{
    if(node==NULL)
    {
        return;
    }
    Inorder(node->left);
    cout<<node->data<<" ";
    Inorder(node->right);
}

void Preorder(treeNode *node)
{
    if(node==NULL)
    {
        return;
    }
    cout<<node->data<<" ";
    Preorder(node->left);
    Preorder(node->right);
}

void Postorder(treeNode *node)
{
    if(node==NULL)
    {
        return;
    }
    Postorder(node->left);
    Postorder(node->right);
```

```
cout<<node->data<<" ";
}

int main()
{
    treeNode *root = NULL,*temp;
    int ch;
    //clrscr();
    while(1)
    {
        cout<<"\n1.Insert\n2.Inorder\n3.Preorder\n4.Postorder\n5.FindMin\n6.FindMax\n7.Search\n8.Exit\n";
        cout<<"Select the option :";
        cin>>ch;
        switch(ch)
        {
            cout<<"Welcome to BST : "<<endl;
        case 1:
            cout<<"\nEnter element to be insert:";
            cin>>ch;
            root = Insert(root, ch);
            cout<<"\nElements in BST are:";
            Inorder(root);
            break;
        case 2:
            cout<<"\nInorder Travesals is:";
            Inorder(root);
            break;
        case 3:
            cout<<"\nPreorder Traversals is:";
```

```
Preorder(root);
break;
case 4:
    cout<<"\nPostorder Traversals is:";
    Postorder(root);
    break;
case 5:
    temp = FindMin(root);
    cout<<"\nMinimum element is :"<<temp->data;
    break;
case 6:
    temp = FindMax(root);
    cout<<"\nMaximum element is :"<<temp->data;
    break;
case 7:
    cout<<"\nEnter element to be searched:";
    cin>>ch;
    temp = Find(root,ch);
    if(temp==NULL)
    {
        cout<<"Element is not foundn";
    }
    else
    {
        cout<<"Element "<<temp->data<<" is Found\n";
    }
    break;
case 8:
    exit(0);
    break;
default:
    cout<<"\nEnter correct choice:";
    break;
```

```
}  
}  
return 0;  
}
```

OUTPUT:

```
Elements in BST are:0 1 4 6 8 9  
1.Insert  
2.Inorder  
3.Preorder  
4.Postorder  
5.FindMin  
6.FindMax  
7.Search  
8.Exit  
Select the option :2  
  
Inorder Travesals is:0 1 4 6 8 9  
1.Insert
```

```
1.Insert  
2.Inorder  
3.Preorder  
4.Postorder  
5.FindMin  
6.FindMax  
7.Search  
8.Exit  
Select the option :3  
  
Preorder Traversals is:8 0 1 4 6 9
```



```
1.Insert
2.Inorder
3.Preorder
4.Postorder
5.FindMin
6.FindMax
7.Search
8.Exit
Select the option :4

Postorder Traversals is:6 4 1 0 9 8
```

```
1.Insert
2.Inorder
3.Preorder
4.Postorder
5.FindMin
6.FindMax
7.Search
8.Exit
Select the option :5

Minimum element is :0
```

```
1.Insert  
2.Inorder  
3.Preorder  
4.Postorder  
5.FindMin  
6.FindMax  
7.Search  
8.Exit  
Select the option :6
```

CONCLUSION:

In this practical, we learned the concept of Binary Search Tree.

PRACTICAL-11

NO	CONTENT
11	<p><u>AIM:</u></p> <p>Write a program that enters vertices, edges of a Graph and display sequence of vertices to traverse the graph in Depth First Search method</p> <p><u>PROGRAM CODE:</u></p> <pre> #include<iostream> using namespace std; int cost[10][10],i,j,k,n,stk[10],top,v,visit[10],visited[10]; int main() { int m; cout <<"Enter no of vertices:"; cin >> n; cout <<"Enter no of edges:"; cin >> m; cout <<"\nEDGES \n"; for(k=1; k<=m; k++) { cin >>i>>j; cost[i][j]=1; } cout <<"Enter initial vertex to traverse from:"; cin >>v; cout <<"DFS ORDER OF VISITED VERTICES:"; cout << v <<" "; visited[v]=1; </pre>

```
k=1;

while(k<n)
{
    for(j=n; j>=1; j--)
        if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1)
        {
            visit[j]=1;
            stk[top]=j;
            top++;
        }
    v=stk[--top];
    cout<<v << " ";
    k++;
    visit[v]=0;
    visited[v]=1;
}

cout<<"\nPARTH PATEL\n19DCS098"<<endl;
return 0;
}
```

OUTPUT:

```
Enter no of vertices:4
Enter no of edges:4

EDGES
1 2
1 3
2 4
3 4
Enter initial vertex to traverse from:1
DFS ORDER OF VISITED VERTICES:1 2 4 3
PARTH PATEL
19DCS098
```

CONCLUSION:

In this practical, we learned the concept of depth first search.

PRACTICAL-12

NO	CONTENT
12	<p><u>AIM:</u></p> <p>An array of 10 elements and hash function is $(x \bmod 10)$. Handle Collisions using Separate Chaining. Keys: 75, 66, 42, 192, 91, 40, 49, 87, 67, 16, 417, 130, 372, 227 Write a program display the final values.</p> <p><u>PROGRAM CODE:</u></p> <pre> #include<iostream> #include<list> using namespace std; class Hash { int BUCKET; list<int> *table; public: Hash(int V); void insertItem(int x); void deleteItem(int key); int hashFunction(int x) { return (x % BUCKET); } void displayHash(); }; Hash::Hash(int b) { this->BUCKET = b; table = new list<int>[BUCKET]; </pre>

```
}

void Hash::insertItem(int key)
{
    int index = hashFunction(key);
    table[index].push_back(key);
}

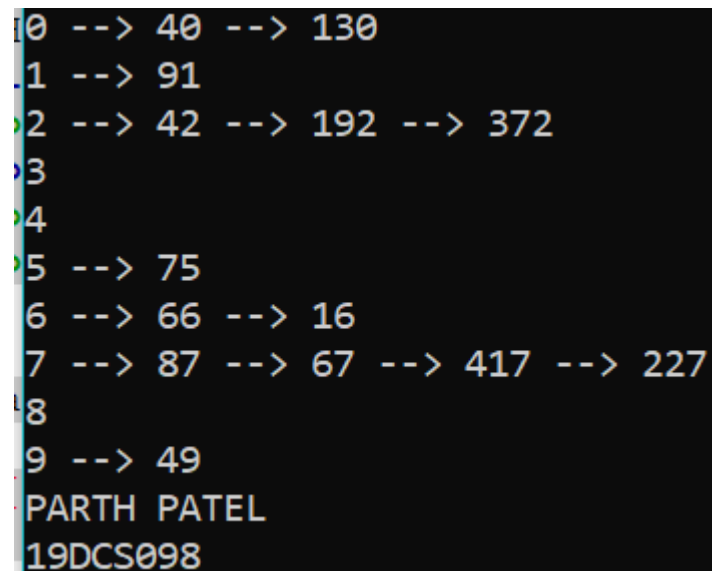
void Hash::deleteItem(int key)
{
    int index = hashFunction(key);
    list<int> :: iterator i;
    for (i = table[index].begin();
         i != table[index].end(); i++) {
        if (*i == key)
            break;
    }
    if (i != table[index].end())
        table[index].erase(i);
}

void Hash::displayHash() {
    for (int i = 0; i < BUCKET; i++) {
        cout << i;
        for (auto x : table[i])
            cout << " --> " << x;
        cout << endl;
    }
}

int main()
{
    int a[] = {75, 66, 42, 192, 91, 40, 49, 87, 67, 16, 417, 130, 372, 227};
    int n = sizeof(a)/sizeof(a[0]);
```

```
Hash h(10);  
for (int i = 0; i < n; i++)  
    h.insertItem(a[i]);  
h.deleteItem(12);  
h.displayHash();  
cout<<"PARTH PATEL\n19DCS098"<<endl;  
return 0;  
}
```

OUTPUT:



```
0 --> 40 --> 130  
1 --> 91  
2 --> 42 --> 192 --> 372  
3  
4  
5 --> 75  
6 --> 66 --> 16  
7 --> 87 --> 67 --> 417 --> 227  
8  
9 --> 49  
PARTH PATEL  
19DCS098
```

CONCLUSION:

In this practical, we learned about Hash Maps