Data Structure

1. Write a Program to implement BubbleSort.

```
 #include<iostream.h>

#include<conio.h>

#include<process.h>


class bubble
{
int *a;

public:
int n;
void get();
void disp();
void bubbleSort();

};
void bubble::get()
{
cout<<"Enter the length of array"<<endl;
cin>>n;
a=new int[n];
cout<<"Enter "<<n<<" elements"<<endl;
for(int i=0;i<n;i++)
{
cin>>a[i];
}
}
void bubble::disp()
{
for(int i=0;i<n;i++)
{
cout<<a[i]<<" ";
}
cout<<endl;
}
```

```
void bubble::bubbleSort()
{
for(int i=0;i<n-1;i++)
{
for(int j=0;j<n-1;j++)
{
   if(a[j]>a[j+1])
   {
   // swap
   int temp=a[j];
   a[j]=a[j+1];
   a[j+1]=temp;
   }
}
}
}
int main()
{
bubble obj;
clrscr();
obj.get();
obj.bubbleSort();
obj.disp();
getch();
return 0;
}
/*
Enter the length of array
5
Enter 5 elements
5 4 3 2 1
1 2 3 4 5
*/
```

2. Write a program to implement Quick Sort .

```
#include<iostream.h>
#include<conio.h>
#include<process.h>

class quick
{
int *a;

public:
int n;
void get();
void disp();
int partition(int low,int high,int pivot);
void quickSort(int,int);
};
void quick::get()
{
cout<<"Enter the length of array"<<endl;
cin>>n;
a=new int[n];
cout<<"Enter "<<n<<" elements"<<endl;
for(int i=0;i<n;i++)
{
cin>>a[i];
}
}
```

```
void quick::disp()
{
for(int i=0;i<n;i++)
{
cout<<a[i]<<" ";
}
cout<<endl;
}
int quick::partition(int low,int high,int pivot)
{
int i=low;
int j=low;
while(i<=high)
{
if(a[i]>pivot)
{    i++;

}else{
int temp=a[i];
a[i]=a[j];
a[j]=temp;
i++;
j++;
}

}
//cout<<j-1<<endl;
return j-1;
}
void quick::quickSort(int low,int high)
{
 if(low>=high)
 {
 return;
 }
 int pi=partition(low,high,a[high]);
quickSort(low,pi-1);
 quickSort(pi+1,high);
}
int main()
{
clrscr();
//cout<<"Hii this is Harshal Sindhi";
quick q1;
q1.get();
q1.quickSort(0,q1.n - 1);
q1.disp();
getch();
return 0;
}

/*
Enter the length of array
8
Enter 8 elements
8 7 6 5 4 3 2 1
output:
1 2 3 4 5 6 7 8
*/
```

3. Write a program to implement Selection Sort

```
#include<iostream.h>
#include<conio.h>
#include<process.h>

class selection
{
int *a;

public:
int n;
void get();
void disp();
void selectionSort();

};
void selection::get()
{
cout<<"Enter the length of array"<<endl;
cin>>n;
a=new int[n];
cout<<"Enter "<<n<<" elements"<<endl;
for(int i=0;i<n;i++)
{
cin>>a[i];
}
}
void selection::disp()
{
```

```
for(int i=0;i<n;i++)
{
cout<<a[i]<<" ";
}
cout<<endl;
}

void selection::selectionSort()
{
for(int i=0;i<n-1;i++)
{
int min=i;
for(int j=i+1;j<n;j++){
if(a[j]<a[min])
{
min=j;
}
}
int temp=a[i];
a[i]=a[min];
a[min]=temp;
}
}
int main()
{
selection obj;
clrscr();
obj.get();
obj.selectionSort();
obj.disp();
getch();
return 0;
}
/*
Enter the length of array
5
Enter 5 elements
5 4 3 2 1
1 2 3 4 5
*/
```

4. Write a program to implement Insertion Sort

```
#include<iostream.h>
#include<conio.h>
#include<process.h>

class insertion
{
int *a;

public:
int n;
void get();
void disp();
void insertionSort();

};
void insertion::get()
{
cout<<"Enter the length of array"<<endl;
cin>>n;
a=new int[n];
cout<<"Enter "<<n<<" elements"<<endl;
for(int i=0;i<n;i++)
{
cin>>a[i];
}
}
void insertion::disp()
{
```

```
for(int i=0;i<n;i++)
{
cout<<a[i]<<" ";
}
cout<<endl;
}

void insertion::insertionSort()
{
for(int i=1;i<n;i++){
while(i-1 >= 0 && a[i-1]>a[i])
{
int temp=a[i];
a[i]=a[i-1];
a[i-1]=temp;
i--;
}
}
}
int main()
{
insertion obj;
clrscr();
obj.get();
obj.insertionSort();
obj.disp();
getch();
return 0;
}
/*
Enter the length of array
5
Enter 5 elements
5 4 3 2 1
1 2 3 4 5
*/
```

5. Write a program to implement Linear Search

```
#include<iostream.h>
#include<conio.h>
#include<process.h>

class linear
{
int *a;
int ele;
public:
int n;
void get();

void linearSearch();

};
void linear::get()
{
cout<<"Enter the length of array"<<endl;
cin>>n;
a=new int[n];
cout<<"Enter "<<n<<" elements"<<endl;
for(int i=0;i<n;i++)
{
cin>>a[i];
}
cout<<endl;
cout<<"Enter element to be search in element"<<endl;
cin>>ele;
}
```

```
void linear::linearSearch()
{
int flag=0;
for(int i=0;i<n;i++)
{

if(a[i]==ele)
{
flag=1;
cout<<"Element is fount at index "<<i<<endl;

}
}
if(flag==0)
{
cout<<"Element not found in the array"<<endl;
}
}
int main()
{
linear obj;
clrscr();
obj.get();
obj.linearSearch();

getch();
return 0;
}
/*
Enter the length of array
5
Enter 5 elements
1 3 5 7 9

Enter element to be search in element
3
Element is fount at index 1
*/
```

6. Write a Program to implement Binary Search

```
#include<iostream.h>
#include<conio.h>
#include<process.h>

class binary
{
int *a;
int ele;
public:
int n;
void get();
void sort();
void binarySearch();
void disp();
};
void binary::get()
{
cout<<"Enter the length of array"<<endl;
cin>>n;
a=new int[n];
cout<<"Enter "<<n<<" elements"<<endl;
for(int i=0;i<n;i++)
{
cin>>a[i];
}
cout<<endl;
cout<<"Enter element to be search in element"<<endl;
cin>>ele;
}
```

```cpp
void binary::sort()
{
//for sorting used insertion sort
for(int i=1;i<n;i++)
{
while(i-1>=0 && a[i-1]>a[i])
{
int temp=a[i];
a[i]=a[i-1];
a[i-1]=temp;
i--;
}
}
}
void binary::disp()
{
for(int i=0;i<n;i++)
{
cout<<a[i]<<" ";
}
cout<<endl;
}
void binary::binarySearch()
{
int flag=0;
int lo=0;
int hi=n-1;
while(lo<=hi)
{
int mid=(lo+hi)/2;
if(a[mid]==ele){
cout<<"Element is found "<<endl;
return;
}
else if(a[mid]>ele)
{
hi=mid-1;
}else if(a[mid]<ele)
{
lo=mid+1;
}
}
if(flag==0)
{
cout<<"Element not found in the array"<<endl;
}
}
int main()
{
binary obj;
clrscr();
obj.get();
obj.sort();
obj.binarySearch();

getch();
return 0;
}
/*
Enter the length of array
5
Enter 5 elements
5 4 3 2 1

Enter element to be search in element
1
Element is found
*/
```

7. Write a program to implement Stack Operations: Push, pop, display.

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
class stack{
int a[10];
int n;
int top;
public:
stack(){
n=10;
top=-1;

}
void push(int data);
int pop();
void display();
};
void stack::push(int data)
{
if(top==n-1)
{
cout<<"Stack is Overflow"<<endl;
return;
} else{
top++;
a[top]=data;
}
}
```

```
int stack::pop()
{
if(top==-1)
{
cout<<"Stack is Underflow"<<endl;
return -1;
} else{
int ele=a[top];
top--;
return ele;
}
}
void stack::display()
{
for(int i=top;i>=0;i--)
{
cout<<a[i]<<" "<<endl;
}
}
int main()
{
stack obj;
clrscr();
obj.push(1);
obj.push(2);
obj.push(3);
obj.push(4);
obj.push(5);
// 5 4 3 2 1
obj.pop();
// 4 3 2 1
obj.display();
getch();
return 0;
}
/*
Output:
4
3
2
1
```

```
*/
```

8. Write a program to implement linear queue operations insert , delete ,display.

```
#include<iostream.h>
#include<conio.h>
class Queue
{
private:
int front;
int rear;
int* arr;
int n;
public:
Queue()
{
front=-1;
rear=-1;
arr=new int[5];
n=5;
}
int isFull()
{
if(front==0 && rear==n-1)
{
return 1;
}else{
return 0;
}
}
int isEmpty()
{
if(front==-1)
{
 return 1;
 //queue is empty
}
return 0;
}
void insert(int data)
{
if(isFull())
{
cout<<"Queue is overflow";
return;
}
if(front==-1)
{
front=0;
}
rear++;
arr[rear]=data;
}
void del()
{
if(isEmpty())
{
cout<<"Queue is Underflow"<<endl;
return;
}
front++;
}
void display()
{
for(int i=front;i<=rear;i++)
{
cout<<arr[i]<<" ";
}
}

};
int main()
{
Queue q;
clrscr();
cout<<"inserting three elements"<<endl;
q.insert(1);
q.insert(2);
q.insert(3);
q.display();
cout<<endl;
cout<<"Deleting element"<<endl;
q.del();
q.display();
getch();
return 0;
}
```

```
inserting three elements
1 2 3
Deleting element
2 3
```

9. Write a program to implement singly linked list with operations. i)create ii) insert iii) delete

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
class node
{
public:
int data;
node* next;

node(int dt)
```

```
{
 data=dt;
 next=NULL;
}
};
class linkedList
{
node* head;
node* tail;
public:
linkedList()
{
head=NULL;
tail=NULL;
}
void insertAtBegin(int data);
void insertAtEnd(int data);
void deleteAtBegin();
void deleteAtEnd();
void disp();
};
void linkedList::insertAtBegin(int data)
{
 node* n1=new node(data);
 if(head==NULL)
 {
  head=n1;
  tail=n1;

 }else{
 n1->next=head;
 head=n1;
 }
}
void linkedList::insertAtEnd(int data)
{
 node* n1=new node(data);
 if(head==NULL)
 {
  head=n1;
  tail=n1;
 }else{
```

```
 tail->next=n1;
 tail=n1;
 }
}
void linkedList::deleteAtBegin()
{
if(head==NULL)
{
cout<<"List is Empty"<<endl;
}else{
head=head->next;
}
}
void linkedList::deleteAtEnd()
{
if(head==NULL)
{
cout<<"List is Empty"<<endl;
}else{
node* temp=head;
while(temp->next!=tail)
{
temp=temp->next;
}
tail=temp;
tail->next=NULL;
}

}
void linkedList::disp()
{
node* temp=head;
while(temp!=NULL)
{
cout<<temp->data<<" ";
temp=temp->next;

}
cout<<endl;
}
int main(){
linkedList l1;
```

```
clrscr();
l1.insertAtBegin(5);
l1.insertAtBegin(4);
l1.insertAtEnd(6);
l1.insertAtEnd(7);
l1.insertAtEnd(8);
l1.insertAtEnd(9);
//4 5 6 7
l1.deleteAtBegin();
l1.deleteAtBegin();
//6 7
l1.deleteAtEnd();
l1.disp();

getch();
return 0;
}
//output
// 6 7 8
```