1. STACK USING ARRARY

```
#include<iostream>
using namespace std;
template <class T>
class stack
{
	public:
		int stacksize,top;
		T *astack;
stack(int size)
{
	stacksize=size;
	astack=new T[stacksize];
	top=-1;
}
int isempty()
{
	if(top==-1)
	{
		return 1;
	}
	else
	{
		return 0;
	}
}
int isfull()
{
```

```
        if(top==stacksize-1)
        {
                return 1;
        }
        else
        {
                return 0;
        }
}
void push(T item)
{
        if(isfull())
        {
                cout<<"stack is full\n";
        }
        else
        {
                top=top+1;
                astack[top]=item;
        }
}
T pop()
{
        if(isempty())
        {
                cout<<"stack is empty\n";
        }
        else
```

```
        {
                T val;
                val=astack[top];
                top=top-1;
                cout<<"the value is\t"<<val<<"\n";
        }
}
};
int main()
{
        int ch,val;
        stack<int>s1(5);
        do
        {

        cout<<"1-push\n2-pop\n3-exit\n";
        cout<<"enter your choice\n";
        cin>>ch;
        switch(ch)
        {
                case 1:cout<<"enter the value\n";
                        cin>>val;
                        s1.push(val);
                        break;
                case 2:
                        s1.pop();
                        break;
```

```
        }
        }while(ch!=3);

}
```

2. DECIMAL TO BINARY USING STACK

```
#include<iostream>
using namespace std;
template<class T>
class Stack
{
public:
        int stacksize,top;
        T *astack;
Stack(int size)
{
        stacksize=size;
        astack=new T[stacksize];
        top=-1;
}
int isEmpty()
{
        if(top==-1)
        {
                return 1;
        }
```

```
        else
        {
                return 0;
        }
}
int isFull()
{
        if(top==stacksize-1)
        {
                return 1;
        }
        else
        {
                return 0;
        }
}
void push(T item)
{
        if(isFull())
        {
                cout<<"stack is full\n";
        }
        else
        {
                top=top+1;
                astack[top]=item;
        }
}
```

```
T pop()
{
	if(isEmpty())
	{
		cout<<"stack is empty\n";
	}
	else
	{
		T val;
		val=astack[top];
		top=top-1;
		cout<<val<<"\n";
	}
}
};
int main()
{
	Stack<int>s(20);
	int n,r,m,p;
	cout<<"enter the decimal value\n";
	cin>>n;
	m=n;
	cout<<"the decimal value is\t"<<m<<"\n";
	while(n>1)
	{
		r=n%2;
		s.push(r);
		n=n/2;
```

```
            p=n;
        }

        s.push(p);
        cout<<"the binary value is\n";
        while(!s.isEmpty())
        {
            s.pop();
        }
}
```

3. INFIX TO POSTFIX USING STACK

```
#include<iostream>
using namespace std;
template<class T>
class Stack
{
        int stacksize,top;
        T *astack;
public:
        Stack(int size)
        {
            stacksize=size;
            astack=new T[stacksize];
            top=-1;
        }
        int isEmpty()
```

```
{
        if(top==-1)
        {
                return 1;
        }
        else
        {
                return 0;
        }
}
int isFull()
{
        if(top==stacksize-1)
        {
                return 1;
        }
        else
        {
                return 0;
        }
}
void push(T item)
{
        if(isFull())
        {
                cout<<"stack is full\n";
        }
        else
```

```
		{
			top=top+1;
			astack[top]=item;
		}
	}
	T pop()
	{
		if(isEmpty())
		{
			cout<<"stack is empty\n";
		}
		else
		{
			T val;
			val=astack[top];
			top=top-1;
			return val;

		}
	}
	T getstacktopval()
	{
		return astack[top];
	}
};
int operatorpre(char op)
{
	int pre;
```

```
	switch(op)
	{
		case '+':
		case '-':
			pre=1;
			break;
		case '*':
		case '/':
			pre=2;
			break;
	}
	return pre;
}
int main()
{
	Stack<char>s(20);
	char infix[20],postfix[20];
	cout<<"enter the infix form\n";
	cin>>infix;
	int i=0;
	int j=0;
	while(infix[i]!='\0')
	{
		if(isalpha(infix[i]))
		{
			postfix[j]=infix[i];
			i++,j++;
		}
```

```
		else
		{
			if(s.isEmpty())
			{
				s.push(infix[i]);
				i++;
			}
			else
			{

	while(((operatorpre(infix[i]))<=(operatorpre(s.getstacktopval())))&&(!
s.isEmpty()))
				{
					postfix[j]=s.pop();
					j++;
				}

				s.push(infix[i]);
				i++;
			}

		}
	}
	while(!s.isEmpty())
	{
		postfix[j]=s.pop();
		j++;
	}
	postfix[j]='\0';
```

```
	cout<<"the infix form is\t"<<infix<<"\n";
	cout<<"the postfix form is\t"<<postfix<<"\n";
}
```

4. IMPLEMENT A QUEUE USING ARRAY

```
#include<iostream>
using namespace std;
template<class T>
class Queue
{
public:
	int front,rear;
	T *que;
	int qsize;
Queue(int size)
{
	qsize=size;
	que=new T[qsize];
	front=rear=-1;
}
int isFull()
{
	if(rear==qsize-1)
	{
		return 1;
	}
	else
```

```
        {
                return 0;
        }
}
int isEmpty()
{
        if(front==-1||front>rear)
        {
                return 1;
        }
        else
        {
                return 0;
        }
}
void Insert(T item)
{
        if(rear==-1)
                front=0;
        if(isFull())
        {
                cout<<"\nQ is full\n";
        }
        else
        {
                rear=rear+1;
                que[rear]=item;
        }
```

```
}
T Delete()
{
        T item;
        if(isEmpty())
        {
        cout<<"Que is empty\n";
        }
        else
        {
                T item;
                item=que[front];
                front=front+1;
                //if(rear=front+1)
                //rear=front=-1;
                //return item;
                cout<<"the value is\t"<<item;

        }
}
};
int main()
{
        int ch,val;
        Queue<int>Q1(3);
        do
        {
                cout<<"\n1-insertion\n2-deletion\n";
```

```
            cout<<"enter your choice\n";
            cin>>ch;
            switch(ch)
            {
                case 1:cout<<"enter the value\n";
                    cin>>val;
                    Q1.Insert(val);
                    break;
                case 2:Q1.Delete();
                    break;
            }
        }while(ch!=3);
}
```

5.IMPLEMENT A CIRCULAR QUEUE

```
#include<iostream>
using namespace std;
template<class T>
class CQ
{
public:
    int size,rear,front;
    T *que;
CQ(int s)
{
    size=s;
    que=new T[size];
```

```
        front=rear=0;
}
T isFull()
{
        if((rear+1)%size==front)
        {
                return 1;
        }
        else
        {
                return 0;
        }
}
int isEmpty()
{
        if(front==rear)
        {
                return 1;
        }
        else
        {
                return 0;
        }
}
void insert(T val)
{
        if(!isFull())
        {
```

```
                rear=(rear+1)%size;
                que[rear]=val;
        }
        else
        {
                cout<<"CQ is full\n";
        }
}
T Delete()
{
        T val;
        if(!isEmpty())
        {
                front=(front+1)%size;
                val=que[front];
                cout<<"the deleted value is\t"<<val<<"\n";
        }
        else
        {
                cout<<"the cq is empty\n";
        }
}
};
int main()
{
        int ch,val;
        CQ<int>c1(3);
        do
```

```
	{
		cout<<"1-insert,2-delete\n";
		cout<<"enter your choice\n";
		cin>>ch;
		switch(ch)
		{
			case 1:
				cout<<"enter the value\n";
				cin>>val;
				c1.insert(val);
				break;
			case 2:
				val=c1.Delete();
				break;
		}
	}while(ch!=3);
}
```

6.IMPLEMENT A LINKED LIST WITH INSERTION,DELETION,DISPLAY AND SEARCH FUNCTIONS

```
#include<iostream>
using namespace std;
template<class T>
class Linklist;
template<class T>
class Node
{
```

```
        T data;
        Node *next;
        friend class Linklist<T>;
};
template<class T>
class Linklist
{
        Node<T> *head;
public:

        Linklist()
        {
                head=NULL;
        }
        void insert(T value)
        {
                Node<T> *cur,*prev;
                if(head==NULL)
                {
                        cur=new Node<T>;
                        cur->data=value;
                        cur->next=NULL;
                        head=cur;
                }
                else
                {
                        cur=head;
                        while(cur!=NULL)
```

```
                {
                        prev=cur;
                        cur=cur->next;
                }

        cur=new Node<T>;
        cur->data=value;
        prev->next=cur;
        }
}
void display()
{
        Node<T>*cur;
        cur=head;
        cout<<"the elements and address is \n";
        while(cur!=NULL)
        {
                cout<<cur->data<<"->";
                cur=cur->next;
        }
        cout<<"NULL\n";
}
int search(T value)
{

        Node<T>*cur;
        cur=head;
        while(cur!=NULL)
```

```
        {
                if(value==cur->data)
                {
                        cout<<"the element is present\n";
                        break;
                }
                else
                {
                        cur=cur->next;
                }

        }if(cur==NULL)
                {
                        cout<<"the element is not present\n";
                        return 0;
                }
}
void insertpos(int pos,T val)
{
        int i=1;
        Node<T>*cur,*prev;
        cur=head;
        prev=NULL;
        while(i<pos&&cur!=NULL)
        {
                prev=cur;
                cur=cur->next;
                i++;
```

```
        }
        if(cur==head)
        {
                cur=new Node<T>;
                cur->data=val;
                cur->next=head;
                head=cur;
        }
        else
        {
                cur=new Node<T>;
                cur->data=val;
                cur->next=prev->next;
                prev->next=cur;
        }
}
void Delete(T val)
{
        Node<T>*cur,*prev;
        cur=head;
        while(cur!=NULL)
        {
                if(cur->data!=val)
                {
                        prev=cur;
                        cur=cur->next;
                }
                else
```

```
                {
                        break;
                }
        }
                if(cur==head)
                {
                        head=cur->next;
                        delete cur;
                }
                else
                {
                        prev->next=cur->next;
                        delete cur;
                }
}
};
int main()
{
        Linklist<int>ob;
        int ch,val,p;
        do
        {
                cout<<"1-insert,2-insertposition,3-search,4-delete,5-display\
n";
                cout<<"enter your choice\n";
                cin>>ch;
                switch(ch)
                {
                case 1:
```

```
                cout<<"enter value\n";
                cin>>val;
                ob.insert(val);
                break;
            case 2:
                cout<<"enter position and value\n";
                cin>>val;
                cin>>p;
                ob.insertpos(val,p);
                break;
            case 3:
                cout<<"enter the searching value\n";
                cin>>val;
                ob.search(val);
                break;
            case 4:
                cout<<"enter the value\n";
                cin>>val;
                ob.Delete(val);
                break;
            case 5:
                cout<<"print the all elements\n";
                ob.display();
                break;
            }
        }while(ch!=6);
}
```

7.Sorting –Quick sort and Bubble sort

```
#include<iostream>
using namespace std;
class Numbers
{

    int *list;
    int n,size;
public:
    Numbers(int s)
    {
        size=s;
        list=new int[size];

    }
    void read()
    {

        cout<<"enter the number of elements";
        cin>>n;
        cout<<"enter the elements:";
        for (int i=0;i<n;i++){
            cin>>list[i];
        }
    }
   void display()
   {
```

```
          cout<<" the elements are:";
        for (int i=0;i<n;i++){
          cout<<list[i]<<"\t";
        }

   }
   void QuickSort(int lb,int ub)
{         int  key,i,j,t;
          if(lb>=ub)    return; // list with less than two elements
          key=list[lb]; i=lb+1; j=ub;
          do
          {
                    while(i<=ub && list[i]<key) i++;
                    while(j>=lb && list[j]>key)  j--;
                    if(i<j)
                    {
                       t=list[i];
                       list[i]=list[j];
                       list[j]=t;

                    }
          }while(i<j);

          t=list[lb];
    list[lb]=list[j];
    list[j]=t;
          QuickSort(lb,j-1);
          QuickSort(j+1,ub);
```

```
}
int getn()
{

    return n;

}

void bubblesort()
{
for(int i=0;i<n-1;i++) //loop for n-1 passes
for(int j=0;j<n-1-i;j++)//loop for comparison and swaping adjacent members
if(list[j]>=list[j+1])
{
list[j]+=list[j+1],
list[j+1]=list[j]-list[j+1],
list[j]=list[j]-list[j+1];        // interchange without using temp variable
}                              // or use a simple interchange function using temp
}




};


int main()
{
```

```
    Numbers n1(10);
    n1.read();
    n1.display();
    cout<<"\n";
    n1.QuickSort(0,n1.getn()-1);
    cout<<"\n";
    n1.display();
    cout<<"\n";
    n1.read();
    n1.display();
    cout<<"\n";
    n1.bubblesort();
    cout<<"\n";
    n1.display();
}
```