Single Link list

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

class node
{
        private:
                int data;
                node *address;
        public:
                void  add(int);
                void display(void);
                int count(void);
                void sort(void);
                void insert(int, int);
                void remove(int);
                void search(int);
};
node *p;

void node::add(int num)
{
        node *q=p;
        if(p==NULL)
        {
                p=new node;
```

```
            p->data=num;
            p->address=NULL;
        }
        else
        {
            while(q->address!=NULL)
            {
                q=q->address;
            }
            q->address=new node;
            q->address->data=num;
            q->address->address=NULL;
        }
}

void node::display(void)
{
        node *q=p;
        if(p==NULL)
        {
            cout<<"\nNo Linked list\n";
        }
        else
        {
            while(q!=NULL)
            {
```

```
                cout<<q->data<<"\t";
                q=q->address;
            }
        }
}

int node::count(void)
{
        node *q=p;
        int i=0;
        if(q==NULL){
            return 0;
        }
        else {
            while(q!=NULL){
                i++;
                q=q->address;
            }
            return i;
        }
}

void node::sort(void)
{
        int temp;
        node *i,*j;
```

```
        for(i=p;i!=NULL;i=i->address)
        {
                for(j=i->address;j!=NULL;j=j->address)
                {
                        if(i->data>j->data)
                        {
                                temp=i->data;
                                i->data=j->data;
                                j->data=temp;
                        }
                }
        }
}

void node::insert(int pos, int num)
{
        node *q=p;
        node *temp;
        int i;

        if(pos==1)
        {
                p=new node;
                p->data=num;
                p->address=q;
        }
```

```
        else if(pos == count()+1)
        {
                add(num);
        }
        else
        {
                for(i=1;i<=pos-2;i++)
                {
                        q=q->address;
                }
                temp=q->address;
                q->address=new node;
                q->address->data=num;
                q->address->address=temp;
        }
}

void node::remove(int pos)
{
        node *q=p;
        node *temp;
        int i;
        if(pos==1)
        {
                p=p->address;
                delete(q);
```

```
        }
        for(i=1; i<=pos-2;i++)
        {
                q=q->address;
        }

        temp=q->address;
        q->address=q->address->address;
        delete(temp);
}

void node::search(int num)
{
        node *q;
        int flag = 0;
        int pos=1;
        for(q=p;q!=NULL;q=q->address)
        {
                if(q->data==num)
                {
                        flag=1;
                        break;
                }
                pos++;
        }
        if(flag==1)
```

```
	{
		cout<<"\nNumber is found at pos: "<<pos<<"\n";
	}
	else
	{
		cout<<"\nNumber is not found.\n"<<endl;
	}
}

void main(void)
{
	clrscr();
	int num,option,pos;
	char ch='y';
	p=NULL;
	node n;
	while(ch=='y')
	{
		cout<<"\nSelect operation you would like to perform ";
		cout<<"\n 1: Add";
		cout<<"\n 2: Display";
		cout<<"\n 3. Count";
		cout<<"\n 4. Insert";
		cout<<"\n 5. Remove";
		cout<<"\n 6. Search";
		cout<<"\n 7. Sort";
```

```
cout<<"\n Enter an option: ";
cin>>option;
switch(option)
{
	case 1:
	{
		cout<<"\n Enter Number:";
		cin>>num;
		n.add(num);
		cout<<"\n Do you want to continue ?";
		break;
	}
	case 2:
	{
		n.display();
		cout<<"\n Do you want to continue ?";
		break;
	}
	case 3:
	{
		num = n.count();
		cout<<"Number of elements are: "<<num;
		cout<<"\n Do you want to continue ?";
		break;
	}
	case 4:
```

```
{
        cout<<"\nEnter position:";
        cin>>pos;
        cout<<"\nEnter number:";
        cin>>num;
        n.insert(pos,num);
        cout<<"\Do you want to continue ?";
        break;
}
case 5:
{
        cout<<"\nEnter position:";
        cin>>pos;
        n.remove(pos);
        cout<<"\nNode Deleted. \n";
        n.display();
        cout<<"\n Do you want to continue ?";
        break;
}
case 6:
{
cout<<"\nEnter number to be searched: ";
cin>>num;
n.search(num);
cout<<"\n Do you want to continue ?";
break;
```

```
                    }
                    case 7:
                    {
                    n.sort();
                    cout<<"\n Do you want to continue ?";
                    break;
                    }
                }
                cin>>ch;
        }
        getch();
}
```

Double Link list

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

class node
{
        private:
                int data;
                node *address;
                node *prev;
        public:
                void  add(int);
                void display(void);
                int count(void);
                void sort(void);
                void insert(int, int);
                void remove(int);
                void search(int);
                void reverse(void);
};
node *p;

void node::add(int num)
{
        node *q=p;
```

```
        if(p==NULL)
        {
                p=new node;
                p->data=num;
                p->address=NULL;
                p->prev=NULL;
        }
        else
        {
                while(q->address!=NULL)
                {
                        q=q->address;
                }
                q->address=new node;
                q->address->data=num;
                q->address->address=NULL;
                q->address->prev=q;
         }
}

void node::display(void)
{
        node *q=p;
        if(p==NULL)
        {
                cout<<"No Linked list";
```

```
        }
        else
        {
                while(q!=NULL)
                {
                        cout<<q->data<<"\t";
                        q=q->address;
                }
        }
}

int node::count(void)
{
        node *q=p;
        int i=0;
        if(q==NULL){
                return 0;
        }
        else {
                while(q!=NULL){
                        i++;
                        q=q->address;
                }
                return i;
        }
}
```

```
void node::sort(void)
{
	int temp;
	node *i,*j;
	for(i=p;i!=NULL;i=i->address)
	{
		for(j=i->address;j!=NULL;j=j->address)
		{
			if(i->data>j->data)
			{
				temp=i->data;
				i->data=j->data;
				j->data=temp;
			}
		}
	}
}

void node::insert(int pos, int num)
{
	node *q=p;
	node *temp;
	int i;

	if(pos==1)
```

```
        {
                p=new node;
                p->data=num;
                p->address=q;
                p->prev=NULL;
                p->address->prev=p;
        }
        else if(pos == count()+1)
        {
                add(num);
        }
        else
        {
                for(i=1;i<=pos-2;i++)
                {
                        q=q->address;
                }
                temp=q->address;
                q->address=new node;
                q->address->data=num;
                q->address->address=temp;
                q->address->prev=q;
                q->address->address->prev=q->address;
        }
}
```

```
void node::remove(int pos)
{
	node *q=p;
	node *temp;
	int i;
	if(pos==1)
	{
		p=p->address;
		delete(q);
		p->prev=NULL;
		return;
	}
	for(i=1; i<=pos-2;i++)
	{
		q=q->address;
	}

	temp=q->address;
	q->address=q->address->address;
	q->address->prev=q;
	delete(temp);
}

void node::search(int num)
{
	node *q;
```

```
        int flag = 0;
        int pos=1;
        for(q=p;q!=NULL;q=q->address)
        {
                if(q->data==num)
                {
                        flag=1;
                        break;
                }
                pos++;
        }
        if(flag==1)
        {
                cout<<"Number is found at pos: "<<pos<<endl;
        }
        else
        {
                cout<<"Number is not found."<<endl;
        }
}
void node::reverse(void)
{
node *q=p;
while(q->address!=NULL)
{
q=q->address;
```

```
}
do
{
cout<<q->data<<"\t";
q=q->prev;
}while(q!=NULL);
}

void main(void)
{
        clrscr();
        int num,option,pos;
        char ch='y';
        p=NULL;
        node n;
        while(ch=='y')
        {
                cout<<"\nSelect operation you would like to perform \n";
                cout<<"\n 1: Add";
                cout<<"\n 2: Display";
                cout<<"\n 3. Count";
                cout<<"\n 4. Insert";
                cout<<"\n 5. Remove";
                cout<<"\n 6. Search";
                cout<<"\n 7. Sort";
                cout<<"\n 8. Reverse";
```

```
cout<<"\n Enter an option: ";
cin>>option;
switch(option)
{
	case 1:
	{
		cout<<"\n Enter Number:";
		cin>>num;
		n.add(num);
		cout<<"\n Do you want to continue ?";
		break;
	}
	case 2:
	{
		n.display();
		cout<<"\n Do you want to continue ?";
		break;
	}
	case 3:
	{
		num = n.count();
		cout<<"Number of elements are: "<<num;
		cout<<"\n Do you want to continue ?";
		break;
	}
	case 4:
```

```
{
        cout<<"\nEnter position:";
        cin>>pos;
        cout<<"\nEnter number:";
        cin>>num;
        n.insert(pos,num);
        cout<<"\nDo you want to continue ?";
        break;
}
case 5:
{
        cout<<"Enter position:";
        cin>>pos;
        n.remove(pos);
        cout<<"Node Deleted. \n";
        n.display();
        cout<<"\n Do you want to continue ?";
        break;
}
case 6:
{
cout<<"Enter number to be searched: ";
cin>>num;
n.search(num);
cout<<"\n Do you want to continue ?";
break;
```

```
                    }
                    case 7:
                    {
                    n.sort();
                    cout<<"\n Do you want to continue ?";
                    break;
                    }
                    case 8:
                    {
                    n.reverse();
                    cout<<"\nDo you want to continue ?";
                    break;
                    }
            }
            cin>>ch;
        }
        getch();
}
```

Circular Link list

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

class node
{
        private:
                int data;
                node *address;
        public:
                void  add(int);
                void display(void);
                int count(void);
                void sort(void);
                void insert(int, int);
                void remove(int);
                void search(int);
};
node *p,*k;

void node::add(int num)
{
        node  *q=p;
        node  *k=p;
        if(p==NULL)
```

```
	{
		p=k=new node;
		p->data=num;
		p->address=p;
	}
	else
	{
		do
		{
			q=q->address;

		}while(q->address!=p);
		q->address=new node;
		q->address->data=num;
		q->address->address=p;
		k=k->address;
	}
}

void node::display(void)
{
	node *q=p;
	if(p==NULL)
	{
		cout<<"\nNo Linked list\n";
	}
```

```
        else
        {
                do
                {
                        cout<<q->data<<"\t";
                        q=q->address;
                }while(q!=p);
        }
}

int node::count(void)
{
        node *q=p;
        int i=0;
        if(q==NULL){
                return 0;
        }
        else {
                do
                {
                        i++;
                        q=q->address;
                }while(q!=p);
                return i;
        }
}
```

```
void node::sort(void)
{
	int temp;
	node *c,*i;
	c=p;
	i=NULL;
	if(p==NULL)
	{
		cout<<"\nList is empty";
	}
	else
	{
		do
		{
			i=c->address;
			while(i!=p)
			{
				if(c->data>i->data)
				{
					temp=c->data;
					c->data=i->data;
					i->data=temp;
				}
				i=i->address;
			}
```

```
                c=c->address;
            }
            while(c->address!=p);
        }
}
void node::insert(int pos, int num)
{
        node *q=p;
        node *temp=p;
        int i;

        if(pos==1)
        {
            p=new node;
            p->data=num;
            p->address=q;
            do
            {
                q=q->address;
            } while(q->address!=temp);
            q->address=p;
        }
        else if(pos == count()+1)
        {
            add(num);
        }
```

```
        else
        {
                for(i=1;i<=pos-2;i++)
                {
                        q=q->address;
                }
                temp=q->address;
                q->address=new node;
                q->address->data=num;
                q->address->address=temp;
        }
}

void node::remove(int pos)
{
        node *q=p;
        node *temp;
        int i;
        if(pos==1)
        {
                p=p->address;
                delete(q);
        }
        for(i=1; i<=pos-2;i++)
        {
                q=q->address;
```

```
        }

        temp=q->address;
        q->address=q->address->address;
        delete(temp);
}

void node::search(int num)
{
        node *q=p;
        int flag = 0;
        int pos=1;
        do
        {
                if(q->data==num)
                {
                        flag=1;
                        break;
                }
                pos++;
                q=q->address;
        }while(q!=p);
        if(flag==1)
        {
                cout<<"\nNumber is found at pos: \n"<<pos<<endl;
        }
```

```
        else
        {
                cout<<"\nNumber is not found.\n"<<endl;
        }
}

void main(void)
{
        clrscr();
        int num,option,pos;
        char ch='y';
        p=NULL;
        node n;
        while(ch=='y')
        {
                cout<<"\nSelect operation you would like to perform \n";
                cout<<"\n 1: Add";
                cout<<"\n 2: Display";
                cout<<"\n 3. Count";
                cout<<"\n 4. Insert";
                cout<<"\n 5. Remove";
                cout<<"\n 6. Search";
                cout<<"\n 7. Sort";
                cout<<"\n Enter an option: ";
                cin>>option;
                switch(option)
```

```
{
        case 1:
        {
                cout<<"\n Enter Number:";
                cin>>num;
                n.add(num);
                cout<<"\n Do you want to continue ?";
                break;
        }
        case 2:
        {
                n.display();
                cout<<"\n Do you want to continue ?";
                break;
        }
        case 3:
        {
                num = n.count();
                cout<<"Number of elements are: "<<num;
                cout<<"\n Do you want to continue ?";
                break;
        }
        case 4:
        {
                cout<<"\nEnter position:";
                cin>>pos;
```

```
        cout<<"\nEnter number:";
        cin>>num;
        n.insert(pos,num);
        cout<<"\nDo you want to continue ?";
        break;
}
case 5:
{
        cout<<"\nEnter position:";
        cin>>pos;
        n.remove(pos);
        cout<<"\nNode Deleted. \n";
        n.display();
        cout<<"\n Do you want to continue ?";
        break;
}
case 6:
{
        cout<<"\nEnter number to be searched: ";
        cin>>num;
        n.search(num);
        cout<<"\n Do you want to continue ?";
        break;
}
case 7:
{
```

```
                        n.sort();
                        cout<<"\n Do you want to continue ?";
                        break;
                    }
                }
                cin>>ch;
            }
            getch();
}
```

Stack

```
#include<iostream.h>
#include<conio.h>
class stack
{
        int data;
        stack *prev;
        stack *next;
        public:
        void push(int);
        int pop();
        void display();
};
stack *top;
stack *bottom;

void stack::display()
{
        stack *q=bottom;
        if((top==NULL)&&(bottom==NULL))
        {
                cout<<"Stack is empty";
        }
        else
        {
                while(q!=NULL)
```

```
                {
                        cout<<q->data<<"\t";
                        q=q->next;
                }
        }
}

void stack::push(int num)
{
        if((bottom==NULL)&&(top==NULL))
        {
                bottom=top=new stack;
                bottom->data=num;
                bottom->prev=NULL;
                bottom->next=NULL;
        }
        else
        {
                top->next=new stack;
                top->next->prev=top;
                top->next->data=num;
                top->next->next=NULL;
                top=top->next;
        }
}
```

```
int stack::pop()
{
        int num;
        if(bottom==NULL && top==NULL)
        {
                return -1;
        }
        num=top->data;
        top=top->prev;
        if(top!=NULL)
        {
                delete(top->next);
                top->next=NULL;
        }
        else
        {
                delete(bottom);
                bottom=NULL;
        }
        return num;
}

void main()
{
        clrscr();
        stack s;
```

```
int num,opt;
char ch='y';
top=NULL;
bottom=NULL;
while(ch=='y')
{
        cout<<"\n1.Push.\n2.Display\n3.Pop\n";
        cout<<"\nEnter  any  option\n";
        cin>>opt;
        switch(opt)
        {
                case 1:
                {
                        cout<<"\nEnter number to insert: \n";
                        cin>>num;
                        s.push(num);
                        break;
                }
                case 2:
                {
                        s.display();
                        break;
                }
                case 3:
                {
                        num=s.pop();
```

```
                        if(num==-1)
                        {
                                cout<<"\nStack is empty\n";
                        }
                        else
                        {
                                cout<<num<<"\t";
                        }
                }
        }
        cout<<"\nDo you want to continue?\n";
        cin>>ch;
    }
    getch();
}
```

Queue

```
#include<iostream.h>
#include<conio.h>
class queue
{
        int data;
        queue *next;
        public:
        void  add(int);
        void display();
        int remov();
};
queue *front;
queue *rear;
void queue::display()
{
        queue *q=front;
        if(front==NULL && rear==NULL)
        {
                cout<<"Queue is empty";
        }
        else
        {
                while(q!=NULL)
                {
                        cout<<q->data<<"\t";
```

```
                q=q->next;
            }
        }
}
void queue::add(int num)
{
        if(front==NULL && rear==NULL)
        {
            front=rear=new queue;
            front->data=num;
            front->next=NULL;
        }
        else
        {
            rear->next=new  queue;
            rear->next->data=num;
            rear->next->next=NULL;
            rear=rear->next;
        }
}
int queue::remov()
{
        queue *temp;
        int result;
        if(front==NULL && rear==NULL)
        {
```

```
            return -1;
        }
        result= front->data;
        temp=front;
        front=front->next;
        if(front!=NULL)
        {
            delete(temp);
        }
        else
        {
            delete(rear);
            rear=NULL;
        }
        return result;
}
void main()
{
        clrscr();
        queue q;
        int num,opt;
        char ch='y';
        while(ch=='y')
        {
            cout<<"\n1.Add\n2.Display\n3.Remove\n";
            cout<<"\nEnter the option: ";
```

```
cin>>opt;
switch(opt)
{
	case 1:
	{
		cout<<"\nEnter the number to insert: ";
		cin>>num;
		q.add(num);
		break;
	}
	case 2:
		q.display();
		break;
	case 3:
	{
		int dnum;
		dnum=q.remov();
		if(dnum==-1)
		{
			cout<<"\nQueue is empty";
		}
		else
		{
			cout<<"\n"<<dnum;
		}
		break;
```

```
                }
            }
            cout<<"Do you want to continue?(y/n): ";
            cin>>ch;
        }
        getch();
}
```

Double Queue

```
#include<iostream.h>
#include<conio.h>
class queue
{
        int data;
        queue *next;
        queue *prev ;
        public:
        void add(int);
        void finsert(int);
        void display();
        int remov();
        int rremov();
};
queue *front;
queue *rear;
void queue::display()
{
        queue *q=front;
        if(front==NULL && rear==NULL)
        {
                cout<<"Queue is empty";
        }
        else
        {
```

```
                while(q!=NULL)
                {
                        cout<<q->data<<"\t";
                        q=q->next;
                }
        }
}
void queue::add(int num)
{
        if(front==NULL && rear==NULL)
        {
                front=rear=new queue;
                front->data=num;
                front->next=NULL;
                front->prev=NULL;
        }
        else
        {
                rear->next=new  queue;
                rear->next->data=num;
                rear->next->prev=rear;
                rear->next->next=NULL;
                rear=rear->next;
        }
}
void queue::finsert(int num)
```

```
{
        queue *temp;
        if(front==NULL && rear==NULL)
        {
                front=rear=new queue;
                front->data=num;
                front->next=NULL;
                front->prev=NULL;
        }
        else
        {
                temp=front;
                front=new queue;
                front->data=num;
                front->prev=NULL;
                front->next=temp;
                front->next->prev=front;
        }
}
int queue::remov()
{
        queue *temp;
        int result;
        if(front==NULL && rear==NULL)
        {
                return -1;
```

```
        }
        result= front->data;
        temp=front;
        front=front->next;
        if(front!=NULL)
        {
                delete(temp);
                front->prev=NULL;
        }
        else
        {
                delete(rear);
                rear=NULL;
        }
        return result;
}

int queue::rremov()
{
        queue *temp;
        int result;
        if(front==NULL && rear==NULL)
        {
                return -1;
        }
        result=rear->data;
```

```
        temp=rear;
        rear=rear->prev;
        if(rear != NULL)
        {
                delete(temp);
                rear->prev=NULL;
        }
        else
        {
                delete(rear);
                rear=NULL;
        }
        return result;
}
void main()
{
        clrscr();
        queue q;
        int num,opt;
        char ch='y';
        while(ch=='y')
        {
                cout<<"\n1.Add\n2.Display\n3.Remove\n4.front
insert\n5.Remove form rear\n";
                cout<<"\nEnter the option: ";
                cin>>opt;
                switch(opt)
```

```
{
        case 1:
        {
                cout<<"\nEnter the number to insert: ";
                cin>>num;
                q.add(num);
                break;
        }
        case 2:
                q.display();
                break;
        case 3:
        {
                int dnum;
                dnum=q.remov();
                if(dnum==-1)
                {
                        cout<<"\nQueue is empty";
                }
                else
                {
                        cout<<"\n"<<dnum;
                }
                break;
        }
        case 4:
```

```
                {
                        cout<<"\nEnter value: \n";
                        cin>>num;
                        q.finsert(num);
                        break;
                }
                case 5:
                {
                        int dnum;
                        dnum=q.rremov();
                        if(dnum==-1)
                        {
                                cout<<"\nQueue is empty";
                        }
                        else
                        {
                                cout<<"\n"<<dnum;
                        }
                        break;
                }
        }
        cout<<"Do you want to continue?(y/n): ";
        cin>>ch;
    }
    getch();
}
```

Priority Queue:

```
#include<iostream.h>
#include<conio.h>
class queue
{
	int data;
	queue *next;
	queue *prev;
	public:
	void add(int);
	void display();
	int remov();
	void prior(void);
};
queue *front;
queue *rear;
void queue::display()
{
	queue *q=front;
	if(front==NULL && rear==NULL)
	{
		cout<<"Queue is empty";
	}
	else
	{
		while(q!=NULL)
		{
			cout<<q->data<<"\t";
			q=q->next;
		}
	}
}
void queue::add(int num)
{
	if(front==NULL && rear==NULL)
	{
		front=rear=new queue;
		front->data=num;
		front->next=NULL;
		front->prev=NULL;
	}
	else
```

```
	{
		rear->next=new queue;
		rear->next->data=num;
		rear->next->next=NULL;
		rear=rear->next;
		rear->next->prev=rear;
	}
}

void queue :: prior(void)
{
	int result;
	int num;
	if((front==NULL)&&(rear==NULL))
	{
		cout<<"Queue is empty";
	}
	else
	{
		if(front->data > rear->data)
		{
			result=remov();
			cout<<"\nData remove = "<<result;
		}
		else
		{
			cout<<"\nEnter number :";
			cin>>num;
			add(num);
		}
	}
}

int queue::remov()
{
	queue *temp;
	int result;
	if(front==NULL && rear==NULL)
	{
		return -1;
	}
	result= front->data;
	temp=front;
```

```
        front=front->next;
        if(front!=NULL)
        {
                delete(temp);
        }
        else
        {
                delete(rear);
                rear=NULL;
        }
        return result;
}
void main()
{
        clrscr();
        queue q;
        int num,opt;
        char ch='y';
        while(ch=='y')
        {
                cout<<"\n1.Add\n2.Display\n3.Remove\n4.Priority\n";
                cout<<"\nEnter the option: ";
                cin>>opt;
                switch(opt)
                {
                        case 1:
                        {
                                cout<<"\nEnter the number to insert: ";
                                cin>>num;
                                q.add(num);
                                break;
                        }
                        case 2:
                                q.display();
                                break;
                        case 3:
                        {
                                int dnum;
                                dnum=q.remov();
                                if(dnum==-1)
                                {
                                        cout<<"\nQueue is empty";
                                }
```

```
                        else
                        {
                                cout<<"\n"<<dnum;
                        }
                        break;
                }
                case 4:
                {
                        q.prior();
                        break;
                }
        }
        cout<<"Do you want to continue?(y/n): ";
        cin>>ch;
    }
    getch();
}
```

Circular Queue:

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

class cqueue
{
	private:
		int data;
		cqueue *next;
	public:
		void add(int);
		void display(void);
		int remov();
};
cqueue *front,*rear;

void cqueue::add(int num)
{
	if(front==NULL && rear==NULL)
	{
		front=rear=new cqueue;
		front->data=num;
		front->next=front;
	}
	else
	{
		rear->next=new cqueue;
		rear->next->data=num;
		rear->next->next=front;
		rear=rear->next;
	}
}

void cqueue::display(void)
{
	cqueue *q=front;
	if(front==NULL && rear==NULL)
	{
		cout<<"\nQueue is empty\n";
	}
	else
```

```
	{
		do
		{
			cout<<q->data<<"\t";
			q=q->next;
		}while(q!=front);
	}
}

int cqueue::remov()
{
	cqueue *temp;
	int result;
	if(front==NULL && rear==NULL)
	{
		return -1;
	}
	result=front->data;
	temp=front;
	front=front->next;
	if(front->next!=front)
	{
		rear->next=front;
		delete(temp);
	}
	else
	{
		delete(rear);
		rear=NULL;
		front=NULL;
	}
	return result;
}

void main(void)
{
	clrscr();
	int num,option;
	char ch='y';
	front=NULL;
	rear=NULL;
	cqueue n;
	while(ch=='y')
```

```
	{
		cout<<"\nSelect operation you would like to perform \n";
		cout<<"\n 1: Add";
		cout<<"\n 2: Display";
		cout<<"\n 3. Remove";
		cout<<"\n Enter an option: ";
		cin>>option;
		switch(option)
		{
			case 1:
			{
				cout<<"\n Enter Number:";
				cin>>num;
				n.add(num);
				cout<<"\n Do you want to continue ?";
				break;
			}
			case 2:
			{
				n.display();
				cout<<"\n Do you want to continue ?";
				break;
			}
			case 3:
			{
			     num=n.remov();
				if(num==-1)
					cout<<"\nempty queue\n";
				else
				{
					cout<<num<<"\n";
				}
				cout<<"\n Do you want to continue ?";
				break;
			}
		}
		cin>>ch;
	}
	getch();
}
```

Stack Using Array:

```
#include<iostream.h>
#include<conio.h>
int max=20;
int top=-1;
int i;
class stack
{
private:
int arrs[20];
public:
void push(void);
void pop(void);
void display(void);
};
void stack::display(void)
{
if(top==-1)
{
cout<<"stack is empty";
}
else
{
for(i=0;i<=top;i++)
{
cout<<arrs[i]<<"\n";
}
}
}
void stack::push(void)
{
int num;
if(top==max-1)
{
cout<<"stack is full";
}
else
{
cout<<"enter the number to push\n";
cin>>num;
top++;
arrs[top]=num;
```

```
}
}
void stack::pop(void)
{
if(top==-1)
{
cout<<"stack is empty/underflow";
}
else
{
for(i=0;i<top;i++)
{
cout<<arrs[top];
arrs[top]='/0';
top--;
}
}
}
void main(void)
{
clrscr();
char ch='y';
int option;
stack s;
while(ch=='y')
{
cout<<"1.display\n";
cout<<"2.Push\n";
cout<<"3.Pop\n";
cout<<"enter option";
cin>>option;
switch(option)
{
case 1:
{
s.display();
cout<<"\ndo you want to continue";
break;
}
case 2:
{
s.push();
cout<<"\ndo you want to continue";
```

```
break;
}
case 3:
{
s.pop();
cout<<"\ndo you want to continue";
break;
}
}
ch=getch();
}
}
```