1A.Insertion sort

```
#include<stdio.h>
#include<conio.h>
void insertion(int a[],int n)
{
	int i,j,temp;
	for(i=0;i<n;i++)
	{
		temp=a[i];
		j=i-1;
		while((temp<a[j])&&(j>=0))
		{
			a[j+1]=a[j];
			j=j-1;
		}
		a[j+1]=temp;
	}
	printf("Elements after sorting\n");
	for(i=0;i<n;i++)
	{
		printf("%d\n",a[i]);
	}
}
void main()
{
	int a[30],n,k;
	clrscr();
	printf("Enter total no. of elements\n");
	scanf("%d",&n);
	printf("Enter the elements\n");
	for(k=0;k<n;k++)
	{
		scanf("%d",&a[k]);
	}
	printf("Elements before sorting\n");
	for(k=0;k<n;k++)
	{
		printf("%d\n",a[k]);
	}
	insertion(a,n);
	getch();
}
```

1B. bubble sort

```
#include<stdio.h>
#include<conio.h>
#define size 10
void bucket_sort(int n,int a[10]);
void main()
{
	int i,n,a[10];
	clrscr();
	printf("Enter the total number of elements\n");
	scanf("%d",&n);
	printf("Enter the array elements\n");
	for(i=0;i<n;i++)
	{
		scanf("%d",&a[i]);
	}
	bucket_sort(n,a);
	printf("The sorted elements are\n");
	for(i=0;i<n;i++)
		printf("%d\n",a[i]);
	getch();
}
void bucket_sort(int n,int a[10])
{
	int i,j,k,b[size];
	for(j=0;j<size;j++)
	{
		b[j]=0;
	}
	for(i=0;i<n;i++)
	{
		b[a[i]]++;
	}
	for(i=0,j=0;j<size;j++)
	{
		for(k=b[j];k>0;k--)
		{
			a[i]=j;
			i=i+1;
		}
	}
}
```

2. stacks

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#define size 5
int top=-1,s[5],item;

void push()
{
	if(top==size-1)
	{
		printf("Stack overflow\n");
		return;
	}
	top=top+1;
	s[top]=item;
}
int pop()
{
	int item_del;
	if(top==-1)
	{
		printf("Stack Empty\n");
		return -1;
	}
	item_del=s[top];
	top=top-1;
	return item_del;
}
void display()
{
	int i;
	if(top==-1)
	{
		printf("Stack Empty\n");
		return;
	}
	for(i=0;i<=top;i++)
	{
		printf("%d\n",s[i]);
	}
}
```

```
void main()
{
	int item_del,choice;
	clrscr();
	for(;;)
	{
		printf("Enter your choice\n 1.push\t 2.pop\t 3.display\n");
		scanf("%d",&choice);
		switch(choice)
		{
			case 1:printf("Enter the Element to be inserted\n");
				scanf("%d",&item);
				push();
				break;
			case 2:item_del=pop();
				if(item_del!=-1)
				printf("Popped Element is %d\n",item_del);
				break;
			case 3:display();
				break;
			default:getch();
				exit(0);
		}
	}
}
```

3. //queues

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#define qsize 3
int q[3],item,front=0,rear=-1;
void insertq()
{
	if(rear==qsize-1)
	{
		printf("Q is Full\n");
		return;
	}
	rear=rear+1;
	q[rear]=item;
}
int deleteq()
{
	int item_del;
	if(front>rear)
	{
		printf("Q Empty\n");
		return -1;
	}
	item_del=q[front];
	front=front+1;
	return item_del;
}
void display()
{
	int i;
	if(front>rear)
	{
		printf("Q Empty\n");
		return;
	}
	printf("Queue Contents\n");
	for(i=front;i<=rear;i++)
	{
		printf("%d\n",q[i]);
	}
}
```

```
void main()
{
	int choice,item_del;
	clrscr();
	for(;;)
	{
		printf("Enter your choice\n 1.insert\t 2.delete\t 3.display\n");
		scanf("%d",&choice);
		switch(choice)
		{
			case 1:printf("Enter your item\n");
				scanf("%d",&item);
				insertq();
				break;
			case 2:item_del=deleteq();
				printf("your Deleted item is\n");
				printf("%d\n",item_del);
				break;
			case 3:display();
				break;
			default:exit(0);
				getch();
		}
	}
}
```

4. //Circular queue

```
#include<stdio.h>
#include<conio.h>
#define qsize 3
int count,f,r=-1,item,q[3];
void insert_rear()
{
	if(count==qsize)
	{
		printf("Queue overflow\n");
		return;
	}
	r=r+1;
	q[r]=item;
	count=count+1;
}
int del_front()
{
	int item_del;
	{
		if(count==0)
		{
			printf("Queue Empty\n");
			return -1;
		}
		item_del=q[f];
		f=(f+1)%qsize;
		count=count-1;
		return item_del;
	}
}
void display()
{
	int i,temp=f;
	if(count==0)
	{
		printf("Queue empty\n");
		return;
	}
	printf("CQ contents\n");
	for(i=1;i<=count;i++)
	{
		printf("%d\n",q[temp]);
		temp=(temp+1)%qsize;
	}
}
void main()
{
	int choice,item_del;
	clrscr();
	for(;;)
```

```
	{
		printf("Enter your choice\n 1.insert\t 2.delete\t 3.display\n");
		scanf("%d",&choice);
		switch(choice)
		{
			case 1:printf("Enter the element to be inserted\n");
				scanf("%d",&item);
				insert_rear();
				break;
			case 2:item_del=del_front();
				if(item_del!=-1);
				printf("The deleted item is %d\n",item_del);
				break;
			case 3:display();
				break;
			default:getch();
				exit(0);
		}
	}
}
```

5. //sparse matrix

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
typedef struct
{
	int row;
	int col;
	int val;
}term;
int k;
void read_sparse_matrix(term a[],int m,int n)
{
	int i,j,item;
	k=1;
	a[0].row=m;
	a[0].col=n;
	printf("Enter the Elements\n");
	for(i=0;i<m;i++)
	{
		for(j=0;j<n;j++)
		{
			scanf("%d",&item);
			if(item==0)
				continue;
			a[k].row=i;
			a[k].col=j;
			a[k].val=item;
			printf("Non zero element stored at location %d is %d\n",k,a[k].val);
			k++;
		}
	}
	a[0].val=k-1;
}

void print_sparse_matrix(term a[])
{
	int p;
	for(p=1;p<k;p++)
		printf("Row=%d\t Column=%d\t value=%d\n",a[p].row,a[p].col,a[p].val);
}
```

```
void search_sparse_matrix(term a[],int item)
{
	int i;
	for(i=1;i<k;i++)
	{
		if(item==a[i].val)
		{
			printf("Search successful\n Element found at position %d\n",i);
			getch();
			exit(0);
		}
	}
	printf("Search Unsuccessful\n");
}
void main()
{
	int m,n,item;
	term a[10];
	clrscr();
	printf("Enter Number of rows and columns\n");
	scanf("%d%d",&m,&n);
	read_sparse_matrix(a,m,n);
	print_sparse_matrix(a);
	printf("Enter Element to be searched\n");
	scanf("%d",&item);
	search_sparse_matrix(a,item);
	getch();
}
```

6. //Infix to postfix conversion

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int F(char symbol)
{
	switch(symbol)
	{
		case '+':
		case '-':return 2;
		case '*':
		case '/':return 4;
		case '^':
		case '$':return 5;
		case '(':return 0;
		case '#':return -1;
		default:return 8;
	}
}
int G(char symbol)
{
	switch(symbol)
	{
		case '+':
		case '-':return 1;
		case '*':
		case '/':return 3;
		case '^':
		case '$':return 6;
		case '(':return 9;
		case ')':return 0;
		default:return 7;
	}
}
void infix_postfix(char infix[],char postfix[])
{
	int i,j,top;
	char s[30],symbol;
	top=-1;
	top=top+1;
	s[top]='#';
	j=0;
	for(i=0;i<strlen(infix);i++)
	{
		symbol=infix[i];
		while(F(s[top])>G(symbol))
		{
			postfix[j]=s[top--];
			j=j+1;
		}
```

```
                if(F(s[top])!=G(symbol))
                {
                        top=top+1;
                        s[top]=symbol;
                }
                else
                {
                        top=top-1;
                }
        }
        while(s[top]!='#')
        {
                postfix[j++]=s[top--];
        }
        postfix[j]='\0';
}
void main()
{
        char infix[20],postfix[20];
        clrscr();
        printf("Enter a valid infix expression\n");
        gets(infix);
        infix_postfix(infix,postfix);
        printf("postfix expression is \n");
        printf("%s\n",postfix);
        getch();
}
```

7. Evalation of postfix experssion

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<ctype.h>
int compute(char symbol,int op1,int op2)
{
	switch(symbol)
	{
		case '+':return op1+op2;
		case '-':return op1-op2;
		case '*':return op1*op2;
		case '/':return op1/op2;
		case '$':
		case '^':return pow(op1,op2);
	}
}
void main()
{
	int s[20],op1,op2,res;
	int i,top;
	char postfix[20],symbol;
	clrscr();
	top=-1;
	printf("Enter the postfix expression\n");
	gets(postfix);
	for(i=0;i<strlen(postfix);i++)
	{
		symbol=postfix[i];
		if(isdigit(symbol))
		{
			top=top+1;
			s[top]=symbol-'0';
		}
		else
		{
			op2=s[top--];
			op1=s[top--];
			res=compute(symbol,op1,op2);
			top=top+1;
			s[top]=res;
		}
	}
	res=s[top--];
	printf("The result is %d\n",res);
	getch();
}
```

8. //Towers of hanoi

```
#include<stdio.h>
#include<conio.h>
int count=0;
void tower(int n,char src,char temp,char des)
{
	if(n==1)
	{
		printf("Move dise %d from %c to %c\n",n,src,des);
		count=count+1;
		return;
	}
	tower(n-1,src,des,temp);
	printf("Move disc %d from %c to %c",n,src,des);
	count=count+1;
	tower(n-1,temp,src,des);
}
void main()
{
	int n;
	clrscr();
	printf("Enter the Number of Disc\n");
	scanf("%d",&n);
	tower(n,'s','T','D');
	printf("Total Number of moves: %d\n",count);
	getch();
}
```

9. //Stack using Linked list

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<alloc.h>
struct node
{
        int info;
        struct node *link;
};
typedef struct node *NODE;

NODE getnode()
{
        NODE x;
        x=(NODE)malloc(sizeof(struct node));
        if(x==NULL)
        {
                printf("Out of memory\n");
                getch();
                exit(0);
        }
        return x;
}
NODE insertfront(int item,NODE first)
{
        NODE temp;
        temp=getnode();
        temp->info=item;
        temp->link=first;
        return temp;
}
NODE deletefront(NODE first)
{
        NODE temp;
        if(first==NULL)
        {
                printf("list is empty cannot delete\n");
                return first;
        }
        temp=first;
        temp=temp->link;
        printf("item deleted=%d\n",first->info);
        free(first);
        first=NULL;
        return temp;
}
```

```
void display(NODE first)
{
	NODE temp;
	if(first==NULL)
	{
		printf("LIST IS EMPTY\n");
		return;
	}
	printf("CONTENTS OF THE SINGLY LINKED LIST\n");
	temp=first;
	while(temp!=NULL)
	{
		printf("%d\n",temp->info);
		temp=temp->link;
	}
	printf("\n");
}
void main()
{
	NODE first=NULL;
	int choice,item;
	clrscr();
	for(;;)
	{
		printf("1:InsertFront 2:DeleteFront 3:Display 4:Quit\n");
		printf("Enter your choice\n");
		scanf("%d",&choice);
		switch(choice)
		{
			case 1:printf("Enter the item to be inserted\n");
				scanf("%d",&item);
				first=insertfront(item,first);
				break;
			case 2:first=deletefront(first);
				break;
			case 3:display(first);
				break;
			default:exit(0);
		}
	}
}
```