DS_25Marks

Slip1_2 : carete singly link list i) create ii) display iii)insert first position iv) Delete node at any position

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>

typedef struct node
{
	int data;
	struct node *next;
}NODE;
NODE * head=NULL;
void create()
{
	int i,n;
	NODE *t;
	printf("\n How Many Nodes you Want to Create : ");
	scanf("%d",&n);
	head=(NODE *)malloc(sizeof(NODE));
	printf("\n Enter node 1: ");
	scanf("%d",&head->data);
	t=head;
	for(i=1;i<n;i++)
	{
		t->next=(NODE*)malloc(sizeof(NODE));
		t=t->next;
		printf("\n Enter node %d: ",i);
		scanf("%d",&t->data);

	}
	t->next=NULL;
}

void display()
{
	struct node *t;
	for(t=head;t!=NULL;t=t->next)
	{
		printf("\t %d ->",t->data);
	}
}

void insert_last()
{
	struct node *nw,*t;
	nw=(NODE*)malloc(sizeof(NODE));
	printf("\n Enter node for inserting: ");
	scanf("%d",&nw->data);
	for(t=head;t->next!=NULL;t=t->next);
	t->next=nw;
	nw->next=NULL;

}

void delete()
{
	struct node *t,*temp;
	int cnt=0,i,p;
	printf("\n Enter position for delete Data : ");
	scanf("%d",&p);
	for(t=head;t!=NULL;t=t->next)
		cnt++;
	if(p>cnt)
	{
		printf("\n Invalid position ");
	}
	else
	{
		if(p==1)
		{
			temp=head;
			head=head->next;
			free(temp);
		}
		else if(cnt==p)
```

```
                {
                        for(i=1,t=head;i<p-1;i++)
                                t=t->next;
                        temp=t->next;
                        t->next=NULL;
                        free(temp);
                }
                else
                {
                        for(i=1,t=head;i<p-1;i++)
                                t=t->next;
                        temp=t->next;
                        t->next = temp->next;
                        free(temp);
                }
        }
}


main()
{
        int ch;
         clrscr();

        do
        {
                printf("\n MENU \n 1.Create \n 2.Display \n 3.Add at last position \n
4.Delete node \n 5.Exit ");
                printf("\n enter your choice : ");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:   create();
                                   break;
                        case 2:  display();
                                  break;
                        case 3: insert_last();
                                  break;

                        case 4:         delete();
                break;
```

```
                        case 5:break;

                }
        }
        while(ch!=5);
          getch();
}
```

Slip2_1 : Dynamic implemenataion of stack

```
 #include<conio.h>
#include<stdio.h>
typedef struct node
{
        int data;
        struct node *next;
}NODE;
NODE *top;

void init()
{
        top=NULL;
}

int isEmpty()
{
        if(top==NULL)
                return 1;
        else
                return 0;
}


void push(int n)
{
        struct node *nw;
        nw = (struct node*)malloc(sizeof(struct node));
        nw->data = n;
        nw->next=top;
        top=nw;
```

```
}

int pop()
{
	struct node *temp;
	temp=top;
	if(isEmpty())
		printf("Stack is empty");
	else
	{
		int n;
		n=top->data;
		top=top->next;
		free(temp);
		return n;
	}
}


void display()
{
	struct node *t;
	for(t=top;t!=NULL;t=t->next)
		printf("%d\t",t->data);
}


main()
{
	int ch,n;
	init();

	do
	{
		printf("\n 1.push \n 2.pop \n 3.display \n 4.exit");
		printf("\n Enter your choice : ");
		scanf("%d",&ch);

		switch(ch)
		{
```

```
case 1:printf("\n Enter data ");
```

```
                        scanf("%d",&n);
                        push(n);
                        break;
                case 2: printf("\n TOS is : ");
                        printf("\t %d",pop());
                        break;
                case 3: printf("\n Stack elements are : ");
                        display();
                        break;
                case 4:break;
            }
    }while(ch!=4);
}
```

Slip3_1 :Write a 'C' program to create linked list with given number in which data part of each node contains individual digit of the number.
(Ex. Suppose the number is 584 then the nodes of linked list should contain 5, 8, 4.)

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>

typedef struct node
{
    int data;
    struct node *next;
}NODE;
NODE *head=NULL;

void create(int no)
{
    NODE *s;

    if(head==NULL)
    {           head=(NODE *)malloc(sizeof(NODE));
        head->data=no;
        head->next=NULL;
        }
        else
        {
                for(s=head;s->next!=NULL;s=s->next);
                s->next=(NODE*)malloc(sizeof(NODE));
                s=s->next;
                s->data=no;
                s->next=NULL;
        }
}


void display()
{
        NODE *s;
        for(s=head;s!=NULL;s=s->next)
        {       printf("\t %d -> ",s->data);
        }
}

void main()
{
        int no,k;
        int r=0;

        printf("\nEnter The Number ");
        scanf("%d", &no);

        while(no>0)
        {    k=no%10;
                r=r*10+k;
                no=no/10;
        }

        while(r>0)
        {
                k=r%10;
                create(k);
                r=r/10;
        }
```

```
	printf("\n link list is :");
	display();
	 getch();
}
```

Slip 4_2 : static implementation of circular queue i)Insert ii)Delete iii)display iv)Exit

---

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#define SIZE 20

int Q[SIZE];
int F,R;
void init()
{
	F=R=-1;
}

int CQEmpty()
{
	if(F==R)
		return 1;
	else
		return 0;
}


int CQFull()
{
	if(((R+1)%SIZE == F) || (F==-1 && R==SIZE-1))
		return 1;
	else
		return 0;
}


void AddQ(int n)
```

{

```
        if(CQFull())
                printf("\n Queue is Full");
        else
        {
                R=(R+1)%SIZE;
                Q[R]=n;
        }
}


int DelQ()
{
        if(CQEmpty())
        {       printf("\n QUEUE IS EMPTY");
                return -1;
        }
        else
        {
                int n;
                F=(F+1)%SIZE;
                n=Q[F];
                return n;
        }
}


void display()
{
        int i=F+1;
        do
        {
                printf("%d \t",Q[i]);
                i=(i+1)%SIZE;
        }while(i!=(R+1)%SIZE);
}

void main()
{
        int ch,no,i;
         clrscr();
```

```
init();
do
```

```
	{
		printf("\nMenu \n1.Add \n2.Delete \n3.Display\n4.Exit");

		printf("\nEnter the Choice");
		scanf("%d",&ch);

		switch(ch)
		{
			case 1: printf("\n Enter elements : ");
				scanf("%d",&no);
				AddQ(no);
				break;
			case 2: no=DelQ();
				if(no!=-1)
					printf("\n deleted data is : %d ",no);
				break;
			case 3:		printf("\n Elements are : ");
				 display();
				 break;
			case 4:  exit(0);
			default: printf("\n Invalid choice");
		}
	}while(ch!=4);
	   getch();
}
```

Slip5_2 : Addition of two Polynomial using Function

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void initpoly(int z[])
{
	int i;
	for(i=0;i<10;i++)
		z[i]=0;
}
```

```
void add(int a[],int b[],int c[],int n)
{
        int i;
        for(i=n;i>=0;i--)
        {
                c[i]=a[i]+b[i];
        }
}
void accept(int a[],int n)
{
        int i;

        for(i=n;i>=0;i--)
        {
                printf("\nEnter the Co-efficient of a[%d] term :\t",i);
                scanf("%d",&a[i]);
        }
}
void disp(int a[],int n)
{
        int i;
        for(i=n;i>0;i--)
        {
                if(a[i]!=0)
                {
                        printf("%dX^%d+",a[i],i);
                }
        }
        printf("%d",a[i]);
}
int pow(int x,int i)
{
        int k=0;int ans=1;
        for(k=1;k<=i;k++)
        {       ans=ans*x;}
        return ans;
}
int eval(int a[],int n,int x)
{
	int i,ans=0;
	for(i=n;i>=0;i--)
	{
		ans=ans+(a[i]*pow(x,i));
	}
	return ans;
}

void main()
{
	int a[10],b[10],c[10],n,ch,x,k,m;
	 clrscr();
	do
	{
		printf("\n1.Evalution \n2.Addition of Polynomial \n3.Exit");
		switch(ch)
		{
			case 1:
				printf("\nEnter the Order of Polynomial : ");
				scanf("%d",&n);
				accept(a,n);
				printf("\nPolynomial is : ");
				disp(a,n);
				printf("\nEnter the Value for Variable X : ");
				scanf("%d",&x);
				printf("\n Evaluated Value is %d",eval(a,n,x));
				break;
			case 2:
				initpoly(a);
				initpoly(b);
				initpoly(c);
				printf("\nEnter the Order of 1st Polynomial : ");
				scanf("%d",&m);
				accept(a,m);
				printf("\nEnter the order of 2nd Polynomial : ");
				scanf("%d",&n);
				accept(b,n);
				printf("\nPolynomial 1 : ");
				disp(a,m);
				printf("\nPolynomial 2 : ");
```

```
                    disp(b,n);
                    k=m>n ? m:n;
                    add(a,b,c,k);
                    printf("\nResultant polynomial Is : ");
                    disp(c,k);
                    break;
                case 3:
                    exit(0);
            }
        }
        while(ch!=3);
           getch();
}
```

Slip6_2 : carete singly link list i) create ii) display iii) insert at last position iv) search node v) exit

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
typedef struct node
{
        int data;
        struct node *next;
}NODE;
NODE * head=NULL;
void create()
{
        int i,n;
        NODE *t;
        printf("\n How Many Nodes you Want to Create : ");
        scanf("%d",&n);
        head=(NODE *)malloc(sizeof(NODE));
        printf("\n Enter node 1: ");
        scanf("%d",&head->data);
        t=head;
        for(i=1;i<n;i++)
        {
                t->next=(NODE*)malloc(sizeof(NODE));
                t=t->next;
                printf("\n Enter node %d: ",i);
                scanf("%d",&t->data);

        }
        t->next=NULL;
}


void display()
{
        struct node *t;

        for(t=head;t!=NULL;t=t->next)
        {
                printf("\t %d ->",t->data);
        }
}
void insert()
{
        struct node *nw,*t;

        nw=(struct node *)malloc(sizeof(struct node));
        printf("\n Enter node for inserting: ");
        scanf("%d",&nw->data);

        for(t=head;t->next!=NULL;t=t->next);
        t->next=nw;
        nw->next=NULL;

}

void search()
{
        struct node *t;
        int flag=0,sr;
        printf("\n Enter data to be search :");
        scanf("%d",&sr);
```

```
	for(t=head;t!=NULL;t=t->next)
	{
		if(sr==t->data)
		{
			flag=1;
			break;
		}
	}
	if(flag==1)
		printf("\n Data is found \n");
	else
		printf("\n Data is NOT found \n");
}



main()
{
	int ch;

	 clrscr();

	do
	{
		printf("\n MENU \n 1.Create \n 2.Display \n 3.Insert a new node \n
4.Search node \n 5.Exit ");
		printf("\n enter your choice : ");
		scanf("%d",&ch);
		switch(ch)
		{
			case 1: create();
				break;
			case 2:
				display();
				break;
			case 3: insert();
				break;

			case 4:
				search();
```

```
break;
```

```
                        case 5:
                                exit(0);

                }
        }
        while(ch!=5);
          getch();
}
```

Slip7_2 : Union of two link list
Slip9_2 : intersection of two link list

```
#include<stdio.h>

typedef struct node
{
        int data;
        struct node *next;
}NODE;

NODE *alloc(int val)
{
        NODE *temp;
        temp=(NODE *)malloc(sizeof(NODE));
        temp->data=val;
        temp->next=NULL;
        return temp;
}

NODE *getnode()
{
        NODE *temp;
        temp=(NODE *)malloc(sizeof(NODE));
        printf("\nEnter the data: ");
        scanf("%d",&temp->data);
        temp->next=NULL;
        return temp;
}
NODE *search(NODE *list,int val)
```

```
{
        NODE *ptr;
        for(ptr=list;ptr!=NULL && ptr->data!=val;ptr=ptr->next);
        return(ptr);
}
NODE *findlast(NODE *list)
{
        NODE *ptr;
        for(ptr=list;ptr->next!=NULL;ptr=ptr->next);
        return(ptr);
}

NODE *create()
{
        NODE *ptr,*temp,*list=NULL;
        int n,i;
        printf("\n Enter how many nodes : ");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                temp=temp=(NODE *)malloc(sizeof(NODE));
                printf("\nEnter the data: ");
                scanf("%d",&temp->data);
                temp->next=NULL;
                if(list==NULL)
                {
                        list=temp;
                }
                else
                {
                        ptr=findlast(list);
                        ptr->next=temp;
                }
        }

        return(list);
}


NODE *unionlist(NODE *list1,NODE *list2)
```

{

```
	NODE *temp,*ptr1,*ptr2,*list3=NULL;
	int i,val;

	for(ptr1=list1;ptr1!=NULL;ptr1=ptr1->next)
	{	temp=alloc(ptr1->data);
		if(list3==NULL)
			list3=temp;
		else
		{
			ptr2=findlast(list3);
			ptr2->next=temp;
		}
	}

	for(ptr1=list2;ptr1!=NULL;ptr1=ptr1->next)
	{
		ptr2=search(list1,ptr1->data);
		if(ptr2==NULL)
		{
			temp=alloc(ptr1->data);

			ptr2=findlast(list3);
			ptr2->next=temp;
		}
	}
	return(list3);
}

NODE *intersect(NODE *list1,NODE *list2)
{
	NODE *temp,*ptr1,*ptr2,*list3=NULL;

	for(ptr1=list1;ptr1!=NULL;ptr1=ptr1->next)
	{
		ptr2=search(list2,ptr1->data);
		if(ptr2!=NULL)
		{
			temp=alloc(ptr1->data);
			if(list3==NULL)
				list3=temp;
```

else

```
                {
                        ptr2=findlast(list3);
                        ptr2->next=temp;
                }
        }
    }

    return(list3);
}

void display(NODE *list)
{
    NODE *ptr;
    for(ptr=list;ptr!=NULL;ptr=ptr->next)
        printf("%d->",ptr->data);
    printf("NULL");
}

void main()
{
    NODE *list1=NULL,*list2=NULL,*list3=NULL;
    printf("\nCreate first list.");
    list1=create();
    printf("\nCreate second list.");
    list2=create();

    printf("\n Data in first list:  ");
    display(list1);

    printf("\n Data in second list:  ");
    display(list2);

    printf("\n\n Union of two list is: ");
    list3=unionlist(list1,list2);
    if(list3==NULL)
        printf("\nList of union is empty");
    else
        display(list3);

    printf("\n\n Intersection of two list:  ");
```

```
        list3=intersect(list1,list2);
        if(list3==NULL)
                printf("\nThere is no common elements in list1 and list2");
        else
                display(list3);

              getch();
}
```

Slip 8_2 : Static implemention of queue

```
#include<stdio.h>
#include<conio.h>
#define SIZE 20
int F,R;
int Q[SIZE];
void initQ()
{
        F=-1;
        R=-1;
}

int isEmptyQ()
{
        if(F==R)
                return 1;
        else
                return 0;

}

int isFullQ()
{
        if(R==SIZE-1)
                return 1;
        else
                return 0;

}
```

```
void AddQ(int n)
{
	if(isFullQ())
		printf("Queue is full");
	else
	{
		R++;
		Q[R]=n;
	}
}


int DeleteQ()
{
	if(isEmptyQ())
	{	printf("\n Queue is empty");
		return -1;
	}
	else
	{
		int n;
		n=Q[++F];
		return n;
	}


}

void display()
{
	int i=F+1;

	while(i<=R)
	{
		printf("%d\t",Q[i]);
		i++;
	}
}
main()
```

```
{
	int ch,n;
```

```
         clrscr();
        initQ();

        do
        {
                printf("\n 1.Add \n 2.Delete \n 3.Display \n 4.Exit");
                printf("\n Enter your choice : ");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:printf("\n Enter data ");
                               scanf("%d",&n);
                               AddQ(n);
                               break;
                        case 2: n=DeleteQ();
                                if(n!=-1)
                                {printf("\n deleted data is : %d ",n);
                                }
                                break;
                        case 3 :printf("Elements are :");
                                display();
                        case 4:break;
                }
        }while(ch!=4);
         getch();
}
```

Slip 10_2 : Dynamic implemention of queue

```
#include<stdio.h>
#include<conio.h>
typedef struct node
{
        int data;
        struct node *next;
}NODE;
NODE *F,*R;
void initQ()
```

```
{
        F=R=NULL;
}

int isEmptyQ()
{
        if(F==R)
                return 1;
        else
                return 0;

}

void AddQ(int n)
{
        NODE *nw;
        nw=(NODE *)malloc(sizeof(NODE));
        nw->data=n;
        nw->next=NULL;
        if(F==NULL)
                F=R=nw;

        else
        {
                R->next=nw;
                R=nw;
        }
}


int DeleteQ()
{
        NODE *temp;int no;
        if(isEmptyQ())
        {       printf("\n Queue is empty");
                return -1;
        }
        else
        {
                temp=F;
```

```
n
o
=
t
e
m
p
-
>
d
a
t
a
;
```

```
                F=F->next;
                free(temp);
        }
        return no;

}

void display()
{
        NODE *t;
        for(t=F;t!=NULL;t=t->next)
                printf("%d \t",t->data);
}
main()
{
        int ch,n;
         clrscr();
        initQ();

        do
        {
                printf("\n 1.Add \n 2.Delete \n 3.Display \n 4.Exit");
                printf("\n Enter your choice : ");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:printf("\n Enter data ");
                               scanf("%d",&n);
                               AddQ(n);
                               break;
                        case 2: n=DeleteQ();
                                if(n!=-1)
                                {printf("\n deleted data is : %d ",n);
                                }
                                break;
                        case 3 :printf("Elements are :");
                                display();
                        case 4:break;
                        default :printf("\n Invalid Choice");
                }
```

```
	}while(ch!=4);
	 getch();
}
```

Slip13_2 : carete singly link list i) create ii) display iii) insert at any position iv) search node v) exit

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
typedef struct node
{
	int data;
	struct node *next;
}NODE;
NODE * head=NULL;
void create()
{
	int i,n;
	NODE *t;
	printf("\n How Many Nodes you Want to Create : ");
	scanf("%d",&n);
	head=(NODE *)malloc(sizeof(NODE));
	printf("\n Enter node 1: ");
	scanf("%d",&head->data);
	t=head;
	for(i=1;i<n;i++)
	{
		t->next=(NODE*)malloc(sizeof(NODE));
		t=t->next;
		printf("\n Enter node %d: ",i);
		scanf("%d",&t->data);

	}
	t->next=NULL;
}
```

```
void display()
{
	struct node *t;

	for(t=head;t!=NULL;t=t->next)
	{
		printf("\t %d ->",t->data);
	}
}
void insert()
{
	struct node *nw,*t;
	int cnt=0,i,p;
	printf("\n Enter position ");
	scanf("%d",&p);
	for(t=head;t!=NULL;t=t->next)
		cnt++;
	if(p>=cnt+1 )
	{
		printf("\n Invalid position ");
	}
	else
	{
		nw=(struct node *)malloc(sizeof(struct node));
		printf("\n Enter node for inserting: ");
		scanf("%d",&nw->data);
		if(p==1)
		{
			nw->next=head;
			head=nw;
		}
		else if(p==cnt)
		{
			for(t=head;t->next!=NULL;t=t->next);
			t->next=nw;
			nw->next=NULL;
		}

		else
		{
```

```
                        for(i=1,t=head;i<p-1;i++)
                                t=t->next;
                        nw->next = t->next;
                        t->next = nw;
                }
        }

}

void search()
{
        struct node *t;
        int flag=0,sr;
        printf("\n Enter data to be search :");
        scanf("%d",&sr);
        for(t=head;t!=NULL;t=t->next)
        {
                if(sr==t->data)
                {
                        flag=1;
                        break;
                }
        }
        if(flag==1)
                printf("\n Data is found \n");
        else
                printf("\n Data is NOT found \n");
}



main()
{
        int ch;

         clrscr();

        do
        {
```

```
		printf("\n MENU \n 1.Create \n 2.Display \n 3.Insert a new node \n
4.Search node \n 5.Exit ");
		printf("\n enter your choice : ");
		scanf("%d",&ch);
		switch(ch)
		{
			case 1: create();
				break;
			case 2:
				display();
				break;
			case 3: insert();
				break;

			case 4:
				search();
				break;
			case 5:
				exit(0);

		}
	}
	while(ch!=5);
	  getch();
}
```

Slip16_2 : carete singly link list i) create ii) display iii)insert first position iv) Delete node at any position

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>

typedef struct node
{
	int data;
	struct node *next;
}NODE;
```

```
NODE * head=NULL;
void create()
{
        int i,n;
        NODE *t;
        printf("\n How Many Nodes you Want to Create : ");
        scanf("%d",&n);
        head=(NODE *)malloc(sizeof(NODE));
        printf("\n Enter node 1: ");
        scanf("%d",&head->data);
        t=head;
        for(i=1;i<n;i++)
        {
                t->next=(NODE*)malloc(sizeof(NODE));
                t=t->next;
                printf("\n Enter node %d: ",i);
                scanf("%d",&t->data);

        }
        t->next=NULL;
}

void display()
{
        struct node *t;

        for(t=head;t!=NULL;t=t->next)
        {
                printf("\t %d ->",t->data);
        }
}

void insert_first()
{
        struct node *nw,*t;
        nw=(NODE*)malloc(sizeof(NODE));
        printf("\n Enter node for inserting: ");
        scanf("%d",&nw->data);

        nw->next=head;
        head=nw;;
}

void delete()
{
        struct node *t,*temp;
        int cnt=0,i,p;
        printf("\n Enter position for delete Data : ");
        scanf("%d",&p);
        for(t=head;t!=NULL;t=t->next)
                cnt++;
        if(p>cnt)
        {
                printf("\n Invalid position ");
        }
        else
        {
                if(p==1)
                {
                        temp=head;
                        head=head->next;
                        free(temp);
                }
                else if(cnt==p)
                {
                        for(i=1,t=head;i<p-1;i++)
                                t=t->next;
                        temp=t->next;
                        t->next=NULL;
                        free(temp);
                }
                else
                {
                        for(i=1,t=head;i<p-1;i++)
                                t=t->next;
                        temp=t->next;
                        t->next = temp->next;
                        free(temp);
                }
        }
```

```
}

main()
{
        int ch;
         clrscr();

        do
        {
                printf("\n MENU \n 1.Create \n 2.Display \n 3.Add at first position \n
4.Delete node \n 5.Exit ");
                printf("\n enter your choice : ");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:   create();
                                   break;
                        case 2:  display();
                                  break;
                        case 3: insert_first();
                                  break;

                        case 4:         delete();
                                 break;
                        case 5:break;

                }
        }
        while(ch!=5);
          getch();
}
```

Slip 17_2 : Write a ‘C’ program to create a random array of n integers. Sort the array using bubble sort. Accept a value x from user and use binary search algorithm to check whether the number is present in array or not and output the position if the number is present.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void bsort(int a[],int n)
{
        int temp,i,j;
        for(i=0;i<n;i++)
        {
                for(j=0;j<n-i;j++)
                {
                        if(a[j]>a[j+1])
                        {
                                temp=a[j];
                                a[j]=a[j+1];
                                a[j+1]=temp;
                        }
                }
        } for

}

int binsearch(int a[],int n,int lb,int ub,int num)
{
        int mid;
        while(lb<=ub)
        {
                mid=(lb+ub)/2;
                if(num==a[mid])
                        return mid;
                else if(num<a[mid])
                        ub=mid-1;
                else
                        lb=mid+1;
        }
```

```
return -1;
```

```
}

void main()
{
	int a[20],i,n,p,lb,ub,num;
	 clrscr();
	printf("\n How many number you want to enter : ");
	scanf("%d",&n);

	for(i=0;i<n;i++)
	{	printf("\n Enter number : ");
		scanf("%d",&a[i]);
	}
	printf("\n sorted array is :");
	bsort(a,n);
	for(i=0;i<n;i++)
	{
		printf("\t%d",a[i]);
	}


	printf("\n Enter number you want to search : ");
	scanf("%d",&num);
	lb=0;
	ub=n-1;
	p=binsearch(a,n,lb,ub,num);
	if(p==-1)
		printf("\n Element not found");
	else
		printf("\n Element found");
	 getch();
}
```

Slip18_2 : carete singly link list i) create ii) display iii)insert at in between position iv) Delete node at any position

```
#include<stdio.h>
```

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

typedef struct node
{
	int data;
	struct node *next;
}NODE;
NODE * head=NULL;
void create()
{
	int i,n;
	NODE *t;
	printf("\n How Many Nodes you Want to Create : ");
	scanf("%d",&n);
	head=(NODE *)malloc(sizeof(NODE));
	printf("\n Enter node 1: ");
	scanf("%d",&head->data);
	t=head;
	for(i=1;i<n;i++)
	{
		t->next=(NODE*)malloc(sizeof(NODE));
		t=t->next;
		printf("\n Enter node %d: ",i);
		scanf("%d",&t->data);

	}
	t->next=NULL;
}

void display()
{
	struct node *t;

	for(t=head;t!=NULL;t=t->next)
	{
		printf("\t %d ->",t->data);
	}
}
void insert()
```

```
{
        struct node *nw,*t;

        int cnt=0,i,p;
        printf("\n Enter position ");
        scanf("%d",&p);
        for(t=head;t!=NULL;t=t->next)
                cnt++;
        if(p>=cnt || p==1)
        {
                printf("\n Invalid position ");
        }
        else
        {
                nw=(NODE*)malloc(sizeof(NODE));
                printf("\n Enter node for inserting: ");
                scanf("%d",&nw->data);

                for(i=1,t=head;i<p-1;i++)
                        t=t->next;
                nw->next = t->next;
                t->next = nw;
        }
}

void delete()
{
        struct node *t,*temp;
        int cnt=0,i,p;
        printf("\n Enter position for delete Data : ");
        scanf("%d",&p);
        for(t=head;t!=NULL;t=t->next)
                cnt++;
        if(p>cnt)
        {
                printf("\n Invalid position ");
        }
        else
        {
                if(p==1)
```

```
		{
			temp=head;
			head=head->next;
			free(temp);
		}
		else if(cnt==p)
		{
			for(i=1,t=head;i<p-1;i++)
				t=t->next;
			temp=t->next;
			t->next=NULL;
			free(temp);
		}
		else
		{
			for(i=1,t=head;i<p-1;i++)
				t=t->next;
			temp=t->next;
			t->next = temp->next;
			free(temp);
		}
	}
}


main()
{
	int ch;
	 clrscr();

	do
	{
		printf("\n MENU \n 1.Create \n 2.Display \n 3.Insert node In between 
position \n 4.Delete node \n 5.Exit ");
		printf("\n enter your choice : ");
		scanf("%d",&ch);
		switch(ch)
		{
			case 1:   create();
				  break;
```

```
case 2:  display();
```

```
                                        break;
                              case 3:  insert();
                                        break;

                              case 4:          delete();
                                       break;
                              case 5:break;

                    }
          }
          while(ch!=5);
            getch();
}
```

Slip19_2 : Write a 'C' program to read n integers and create two lists such that all positive numbers are in one list and negative numbers are in another list. Display both the lists in sorted order.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct node
{
          int data;
          struct node *next;
};

struct node* insert(struct node *f,int no)
{
          struct node *t;

          if(f==NULL)
          {                   f=(struct node *)malloc(sizeof(struct node));
                    f->data=no;
                    f->next=NULL;
          }
          else
          {
                for(t=f;t->next!=NULL;t=t->next);
                t->next=(struct node *)malloc(sizeof(struct node));
                t=t->next;
                t->data=no;
                t->next=NULL;
        }

        return f;}
void display(struct node* f)
{
        struct node *t;

        for(t=f;t!=NULL;t=t->next)
        {
                printf("\t %d ->",t->data);
        }
}

void sort(struct node *f)
{
        struct node *p,*q;
        int temp;
        for(p=f;p!=NULL;p=p->next)
        {
                for(q=p->next;q!=NULL;q=q->next)
                {
                        if(p->data > q->data)
                        {
                                temp = p->data;
                                p->data = q->data;
                                q->data = temp;
                        }
                }
        }
}


void main()
{
        struct node* h1=NULL,*h2=NULL;
```

```
	int a[10],n,i;
	printf("\n Enter no of nodes to be created : ");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("\n Enter data : ");
		scanf("%d",&a[i]);

	}
	for(i=0;i<n;i++)
	{
		if(a[i]>0)
		{	 printf("\n condition is true %d \n",a[i]);
			h1=insert(h1,a[i]);
			 display(h1);
		}
		else
		{
			h2=insert(h2,a[i]);
		}
	}

	printf("\n positive data link list is \n");
	display(h1);
	printf("\n Negative data link list is \n");
	display(h2);
	printf("\n Ater sorting postive data link list \n");
	sort(h1);
	display(h1);
	printf("\n Ater sorting negative data link list \n");
	sort(h2);
	display(h2);
	 getch();
}
```

Slip23_2 :convert infix to prefix and evaluate it

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

```
#include<string.h>
#include<stdlib.h>
char st[100];
int top=-1;
int prec(char c)
{
        switch(c)
        {
                case '(':
                        return 0;
                case '+':
                case '-':
                        return 1;
                case '*':
                case '/':
                case '%':
                        return 2;
                case '^':
                        return 3;
        }
        return 0;
}
void push(char c)
{
        top++;
        st[top]=c;
}
char pop()
{
        char c;
        c=st[top];
        top--;
        return c;
}
int main()
{
        int i,j,p,no1,no2;
        char ch,s[30],s1[30],s2[30];
        clrscr();
        printf("\nEnter the Infix Expr");
```

```
gets(s);

s1=strrev(s);
printf("reverse %s",strrev(s));
for(i=0;s1[i]!='\0';i++)
{
        if(s1[i]=='(' )
                s1[i]=')';
        else if(s1[i]== ')' )
                s1[i]='(';
}
j=0;

for(i=0;s1[i]!='\0';i++)
{
        switch(s1[i])
        {
                case '(':
                case '+':
                case '-':
                case '/':
                case '*':
                case '^':
                        p=prec(s1[i]);
                        while(top!=-1 && p<=prec(st[top]))
                                s2[j++]=pop();
                        push(s1[i]);
                        break;
                case ')':
                        do
                        {
                                ch=pop();
                                if(ch!='(')
                                {
                                        s2[j++]=ch;
                                }
                        }

                        while(ch!='(');
                        break;
```

default:

```
                    s2[j++]=s1[i];
        }
}
while(top!=-1)
{
        s2[j++]=pop();
}
s2[j]='\0';
s2=strrev(s2);
printf("\nPreFix Expr %s",strrev(s2));
printf("\n\nEval of PostFix Expr is \n\n");
j=0;
for(i=0;s2[i]!='\0';i++)
{
        if(s2[i]<='9' && s2[i]>='0')
                push(s2[i]-48);
        else
        {
                no2=pop();
                no1=pop();
                switch(s2[i])
                {
                        case '+':
                                push(no1+no2);
                                break;
                        case '-':
                                push(no1-no2);
                                break;

                        case '/': push(no1/no2);
                                  break;

                        case '*':
                                  push(no1*no2);
                                  break;
                        case '^':
                                  push(pow(no1,no2));
                                  break;
                        case '%':
                                  push(no1%no2);
```

```
                                        break;
                            }

                    } end of else
          }
          j=pop();
          printf("Result is %d ",j);
           getch();
}
```

Slip24_2 and Slip26_1: Doubly Circular link list and print it in reverse order

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>

typedef struct node
{
          int data;
          struct node *prev;
          struct node *next;
}NODE;
NODE *head=NULL;
void create()
{
          NODE *t;
          int i,no;
          printf("enter no of nodes");
          scanf("%d",&no);

          head=(NODE *)malloc(sizeof(NODE));
          printf("\n Enter data : ");
          scanf("%d",&head->data);
          head->next=NULL;
          t=head;
          for(i=1;i<no;i++)
          {
                    t->next=(struct node*)malloc(sizeof(struct node));
```

```
                t->next->prev=t;
                t=t->next;
                printf("\n Enter data : ");
                scanf("%d",&t->data);
                t->next=NULL;
        }
        t->next=head;
        head->prev=t;

}

void display()
{
        NODE  *t;
        t=head;
        do
        {
                printf("\t%d",t->data);
                t=t->next;
        }while(t!=head);
}

void rev()
{
        NODE  *s,*t;

        for(s=head;s->next!=head;s=s->next);
        t=s;
        do
        {
                printf("\t%d",t->data);
                t=t->prev;
        }while(t!=head);
        printf("\t%d",t->data);
}

void main()
{
         clrscr();
        create();
```

```
	display();
	printf("\n Reverse Linklist is ");
	rev();
	 getch();
}
```

Slip25_2 :convert infix to postfix and evaluate it

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
char st[100];
int top=-1;
int prec(char c)
{
	switch(c)
	{
		case '(':
			return 0;
		case '+':
		case '-':
			return 1;
		case '*':
		case '/':
		case '%':
			return 2;
		case '^':
			return 3;
	}
	return 0;
}
void push(char c)
{
	top++;
	st[top]=c;
}
char pop()
{
        char c;
        c=st[top];
        top--;
        return c;
}
int main()
{
        int i,j,p,no1,no2;
        char ch,s1[30],s2[30];
           clrscr();
        printf("\nEnter the Infix Expr");
        gets(s1);
        for(i=0;s1[i]!='\0';i++)
        {
                if(s1[i]=='(' )
                        s1[i]=')';
                else if if(s1[i]== ')' )
                        s1[i]='(';
        }
        j=0;

        for(i=0;s1[i]!='\0';i++)
        {
                switch(s1[i])
                {
                        case '(':
                        case '+':
                        case '-':
                        case '/':
                        case '*':
                        case '^':
                                p=prec(s1[i]);
                                while(top!=-1 && p<=prec(st[top]))
                                        s2[j++]=pop();
                                push(s1[i]);
                                break;
                        case ')':
                                do
                                {
                                        ch=pop();
```

```
if(ch!='(')
				{
					s2[j++]=ch;
				}
			}

			while(ch!='(');
			break;
		default:
			s2[j++]=s1[i];
	}
}
while(top!=-1)
{
	s2[j++]=pop();
}
s2[j]='\0';
printf("\nPreFix Expr %s",s2);
printf("\n\nEval of PostFix Expr is \n\n");
j=0;
for(i=0;s2[i]!='\0';i++)
{
	if(s2[i]<='9' && s2[i]>='0')
		push(s2[i]-48);
	else
	{
		no2=pop();
		no1=pop();
		switch(s2[i])
		{
			case '+':
				push(no1+no2);
				break;
			case '-':
				push(no1-no2);
				break;

			case '/': push(no1/no2);
				  break;

			case '*':
```

```
                                    push(no1*no2);
                                    break;
                                    /*case '^':
                                      push(pow(no1,no2));
                                      break;*/
                              case '%':
                                    push(no1%no2);
                                    break;
                         }

                    } end of else
          }

          j=pop();
          printf("Result is %d ",j);
           getch();
}
```

DS 15 Marks

Slip1_1 :Write a 'C' program to reverse a string using Static implementation of Stack.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>
#define SIZE 40
#include<stdio.h>
#include<conio.h>
#define SIZE 20
char stk[SIZE];
int top;
void init()
{  top=-1;  }
int isfull()
{         if(top==SIZE-1)
          return 1;
          else
                    return 0;
}
int isEmpty()
{ if(top==-1)
          return 1;
          else
                    return 0;
}
void push(char c)
{         if(isfull())
          {       printf("Stack is overflow");    }
          else
          {     top++;
                    stk[top]=c;
          }
}
char pop()
{    if(isEmpty())
          {      printf("Stack is empty");  }
          else
          {  char c;
                    c=stk[top];
                    top--;
                    return c;
          }
}
void main()
{       char s[30];
        int i;
        clrscr();
        printf("\nEnter String : ");
        gets(s);
        init();
        for(i=0;s[i]!='\0';i++)
        {       push(s[i]);     }
        printf("\n Reverse String is : ");
        while(!isEmpty())
        {       printf("%c",pop());
        }
        getch();
}
```

Slip 2_1 : Evalution of polynomial

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

int eval(int a[],int n,int x)
{
        int i;
        double p,sum=0;
        for(i=n;i>=0;i--)
        {
                p=pow(x,i);
                sum=sum+a[i]*p;
        }
        return sum;
}

void main()
{
        int a[10],n,c,i,e;
        printf("Enter the Degree of polynomial : ");
        scanf("%d",&n);
        printf("\n Enter the coefficient : ");
        for(i=n;i>=0;i--)
        {
                printf("\n Enter coefficient A[%d]",i);
                scanf("%d",&a[i]);
        }
        printf("\n Enter the polynomial : ");
        for(i=n;i>=0;i--)
        {
                if(a[i]!=0)
                        printf("%dX^%d + ",a[i],i);
        }
        printf("%d",a[i]);

        printf("\n Enter the value for X ");
        scanf("%d",&x);
        e=eval(a,n,x);
        printf("\n Evalution of polynomaial is = %d",e);
        getch();
}
```

Slip3_1 : program for implementing Linear Search method using function.

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

void search(int a[],int n,int k)
{  int flag=0,i;
        for(i=0;i<n;i++)
        {       if(a[i]==k)
                {       flag=1;
                        break;
                }
        }       if(flag==1)
        printf("\nElement is found at %d location ",i+1);
        else
                printf("\nElement is NOT found");
}
void main()
{        int a[20],sr,i,n;
         clrscr();
        printf("\nEnter how many elements : ");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {       printf("\nEnter element : ");
                scanf("%d",&a[i]);
        }
        printf("\nEnter element for search : ");
        scanf("%d",&sr);
        search(a,n,sr);
        getch();
}
```

Slip no4_1 : 2- D mxn matrix using dynamic memory allocation

```
#include<stdio.h>
#include<stdlib.h>

void  main()
{
        int **ptr; int row, col,i, j;

        printf("\nEnter number of rows for matrix : ");
        scanf("%d", &row);
        printf("\nEnter number of columns for matrix : ");
        scanf("%d", &col);


        ptr = (int **) malloc(sizeof(int *) * row);
        ptr[i] = (int *)malloc(sizeof(int) * col);

        printf("\nEnter elements of matrix :\n");
        for(i=0; i< row; i++)
        {
                for(j=0; j< col; j++)
                {
                        printf("\tA[%d][%d] = ",i, j);
                        scanf("%d", &ptr[i][j]);
                }
        }
        printf("\nMatrix is:\n");
        for(i=0; i< row; i++)
        {
                for(j=0; j< col; j++)
                {
                        printf("%d\t", ptr[i][j]);
                }
                printf("\n");
        }
        getch();
}
```

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
typedef struct node
{
	int data;
	struct node *next;
}NODE;
NODE *head=NULL;

void create()
{
	int i,n;
	NODE *t;
	printf("\nHow Many Nodes you Want to Create: ");
	scanf("%d",&n);

	head=(NODE *)malloc(sizeof(NODE));
	printf("\n Enter node 1: ");
	scanf("%d",&head->data);
	t=head;
	for(i=1;i<n;i++)
	{
		t->next=(NODE *)malloc(sizeof(NODE));
		t=t->next;
		printf("\n Enter node %d: ",i);
		scanf("%d",&t->data);
	}
	t->next=NULL;
}

void display()
{
	NODE *t;
	for(t=head;t!=NULL;t=t->next)
	{	printf("\t %d ->",t->data);  }
}
void Count_node()
{	NODE*t;
	int cnt=0;
	for(t=head;t!=NULL;t=t->next)
	{	cnt++;
	}printf("\nNo of nodes are :%d \n ",cnt);
}
```

```
void main()
{ int n;
	 clrscr();
	create();
	printf("\n Link list is : ");
	display();
	Count_node();
      getch();
}
```

Slip6_1 :search given elements into the list using Non-Recursive Binary Search Method.

```
#include<stdio.h>
#include<conio.h>
int b_search(int a[],int lb,int ub,int x)
{	  int mid;
	while(lb<=ub)
	{	mid =(lb+ub)/2;
		if(x==a[mid])
			return 1;
		else if(x<a[mid])
			ub=mid-1;
		else
			lb=mid+1;
	}

	return 0;
}
void main()
{	 int a[20],sr,s,i,n;
	  clrscr();
	printf("\n Enter how many elements : ");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{	 printf("\n Enter Data :");
		scanf("%d",&a[i]);
	}
	printf("\n Enter element to be search : ");
	scanf("%d",&sr);
	s=b_search(a,0,n,sr);
	if(s==1)
		printf("\nElement is found");
	else
		printf("\nElement is NOT found");
	getch();
}
```

Slip7_1 : search given elements into the list using Recursive Binary Search Method.

```
#include<stdio.h>
#include<conio.h>
int b_search(int a[],int lb,int ub,int x)
{ int mid;
        if(lb<=ub)
        {       mid =(lb+ub)/2;
                if(x==a[mid])
                        return 1;
                else if(x<a[mid])
                        return b_search(a,lb,mid-1,x);  search at 1st part
                else
                        return b_search(a,mid+1,ub,x);
        }

        return 0;
}
void main()
{       int a[20],sr,s,i,n;
          clrscr();
        printf("\n Enter how many elements : ");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {        printf("\n Enter Data :");
                scanf("%d",&a[i]);
        }
        printf("\n Enter element to be search : ");
        scanf("%d",&sr);
        s=b_search(a,0,n,sr);
        if(s==1)
                printf("\nElement is found");
        else
                printf("\nElement is NOT found");
        getch();
}
```

Slip8_1 : create 1-D table dynamically

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
```

```
{	int n,i;
```

```
	int *ptr;
	printf("\n Enter no of elements : ");
	scanf("%d",&n);
	ptr=(int *)malloc(n*sizeof(int));
	for(i=0;i<n;i++)
	{	printf("\n Enter no : ");
		scanf("%d",&ptr[i]);
	}
	printf("\n");
	for(i=0;i<n;i++)
	{	printf("%d\t",ptr[i]);
	}
	getch();
}
```

Slip 9_1 : Bubble_sort

```
#include<stdio.h>
#include<conio.h>
void bubblesort(int a[20],int n)
{	int i,j,temp;
	for(i=0;i<n;i++)
	{	for(j=0;j<n-1;j++)
		{	if(a[j]>a[j+1])
			{	temp=a[j];
				a[j]=a[j+1];
				a[j+1]=temp;
			}
		}
	}
}
main()
{	int a[100],i,n;
	clrscr();
	printf("Enter how many elements would you like to enter:");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{	printf("Enter data:");
		scanf("%d",&a[i]);
	}
	bubblesort(a,n);
	printf("The array after sorting is:\n");
	for(i=0;i<n;i++)
		printf("%d\t",a[i]);
	getch();
}
```

Slip10_1 : Create a Singly link list and print it in reverse

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
typedef struct node
{
	int data;
	struct node *next;
}NODE;
NODE *head=NULL;

void create()
{
	int i,n;
	NODE *t;
	printf("\nHow Many Nodes you Want to Create: ");
	scanf("%d",&n);

	head=(NODE *)malloc(sizeof(NODE));
	printf("\n Enter node 1: ");
	scanf("%d",&head->data);
	t=head;
	for(i=1;i<n;i++)
	{
		t->next=(NODE *)malloc(sizeof(NODE));
		t=t->next;
		printf("\n Enter node %d: ",i);
		scanf("%d",&t->data);
	}
	t->next=NULL;
}

void display()
{
	NODE *t;
	for(t=head;t!=NULL;t=t->next)
	{	printf("\t %d ->",t->data);  }
}

void reverse()
```

```
{
        struct node *s,*t;
        int cnt=0,i;
        for(s=head;s!=NULL;s=s->next)
        {cnt++; }
        while(cnt!=0)
        { for(t=head,i=1;i<=cnt-1;i++)
                {  t=t->next;  }
                printf("%d ->\t",t->data);
                cnt--;
        }
}
void main()
{ int n;
    clrscr();
        create();
        printf("\n Link list is : ");
        display();
        printf("\n Reverse Link list is : ");
        reverse();
        getch();
}
```

Slip 11_1 :Accept a string and Reverse each word of String using stack

```
#include<stdio.h>
#include<conio.h>
#define SIZE 20
char stk[SIZE];
int top;
void init()
{  top=-1;  }
        int isfull()
{       if(top==SIZE-1)
        return 1;
        else
                return 0;
}
int isEmpty()
{ if(top==-1)
        return 1;
        else
                return 0;
}
void push(char c)
```

```
{         if(isfull())
          {       printf("Stack is overflow");    }
          else
          {     top++;
                    stk[top]=c;
          }
}
char pop()
{    if(isEmpty())
          {     printf("Stack is empty");  }
          else
          {  char c;
                    c=stk[top];
                    top--;
                    return c;
          }
}
void main()
{         int i;
          char s[20];
                     clrscr();
          printf("\nEnter String : ");
          gets(s);
          init();
          for(i=0;s[i]!='\0'; i++)
          {     if(s[i]==' ')
                    {         while(!isEmpty())
                              {
                                        printf("%c",pop());
                              }
                              printf(" ");
                    }
                    else
                    {          push(s[i]);
                    }
          } end of for
          while(!isEmpty())
          {          printf("%c",pop());
          }
           getch();
}
```

Slip12_1 : create a circular singly link list and display it

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
typedef struct node
{
	int data;
	struct node *next;
}NODE;
NODE *head=NULL;

void create()
{
	int i,n;
	NODE *t;
	printf("\nHow Many Nodes you Want to Create: ");
	scanf("%d",&n);

	head=(NODE *)malloc(sizeof(NODE));
	printf("\n Enter node 1: ");
	scanf("%d",&head->data);
	t=head;
	for(i=1;i<n;i++)
	{
		t->next=(NODE *)malloc(sizeof(NODE));
		t=t->next;
		printf("\n Enter node %d: ",i);
		scanf("%d",&t->data);
	}
	t->next=head;
}

void display()
{
	NODE *t;
	t=head;
	do
	{	printf("\t %d ->",t->data);
		t=t->next;
	} while(t!=head);
}

void main()
{	int n;
	 clrscr();
	printf("\n Circular Link list is : ");
	create();
```

```
        display();
          getch();
}
```

Slip13_1 : sort data using insertion sort

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
void insertion_sort(int a[20],int n)
{       int i,j,temp;
        for(i=1;i<n;i++)
        {       temp=a[i];
                for(j=i-1;j>=0;j--)
                {       if(temp<a[j])
                        {       a[j+1]=         a[j];
                        }
                        else
                                break;
                }
                a[j+1] =temp;
        }
}

main()
{       int a[100],i,n;
                clrscr();

        printf("\n Enter how many elements would you like to enter : ");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {       printf("Enter data:");
                scanf("%d",&a[i]);
        }       insertion_sort(a,n);
        printf("\n The array after sorting is : ");
        for(i=0;i<n;i++)
                printf("%d\t",a[i]);
                getch();
}
```

Slip14_1 : sort data using selection sort

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
void selection_sort(int a[20],int n)
{       int i,j,max,temp,pos;
        for(i=0;i<n;i++)
        {       max=a[0];
                pos=0;
                for(j=1;j<n-i;j++)
                {       if(a[j]>max)
                        {       max=a[j];
                                pos=j;
                        }
                }
                j--;
                temp=a[pos];
                a[pos]=a[j];
                a[j]=max;
        }
}
main()

{       int a[100],i,n;
                clrscr();
        printf("Enter how many elements would you like to enter:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {       printf("Enter data:");
                scanf("%d",&a[i]);
        }
        selection_sort(a,n);

        printf("The array after sorting is:\n");
        for(i=0;i<n;i++)
                printf("%d\t",a[i]);
         getch();
}
```

Slip15_1 : Accept n student names from user and store it in an array. Write a function to search given student name into the array using Linear search method.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
```

```
 #include<conio.h>
void Linear_search(char a[10][10],int n,char s[])
{	int i,flag=0;
	for(i=0;i<n;i++)
	{	if((strcmp(a[i],s))==0)
		{	flag=1;
			break;
		}
	}
	if(flag==1)
		printf("\n %s is found",s);
	else
		printf("\n %s is NOT found",s);
}

void main()
{	char a[10][10];char sr[10];
	int i,n;
		clrscr();
	printf("\n Enter how many no of students : ");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{	printf("\n Enter student name :");
		scanf("%s",a[i]);
	}	printf("\n Array is:\n");
	for(i=0;i<n;i++)
		printf("%s\t",a[i]);
	printf("\n Enter student name to be search : ");
	scanf("%s",sr);
	Linear_search(a,n,sr);
		getch();
}
```

Slip16_1 : Write a 'C' program to accept two polynomials and add these two polynomials using function. Display the result (Use array).

```
#include<stdio.h>
 #include<conio.h>
void add(int m,int a[10],int n,int b[10])
{
	int c[10],i,cnt=0;
	if(m>=n)
	{	for(i=m;i>=0;i--)
		{	c[i]=a[i]+b[i];
			cnt++; }
	}
	else
	{	for(i=n;i>=0;i--)
		{	c[i]=a[i]+b[i];
		}
	}
	printf("\nResultant polynomial is  = ");
	for(i=cnt-1;i>0;i--)
	{	printf("%dX^%d + ",c[i],i);
	}
	printf("%d",c[i]);
}
void main()
{	int a[10],b[10],i,m,n;
	clrscr();
	for(i=0;i<=9;i++)
		a[i]=0;
	for(i=0;i<=9;i++)
		b[i]=0;
	printf("\nEnter the order of first Polynomial");
	scanf("%d",&m);
	printf("\nEnter the Co-efficient");
	for(i=m;i>=0;i--)
	{	scanf("%d",&a[i]);
	}
	printf("\nEnter the order of Second Polynomail");
	scanf("%d",&n);
	printf("\nEnter the Co-efficient");
	for(i=n;i>=0;i--)
	{	scanf("%d",&b[i]);
	}
	add(m,a,n,b);
	getch();
}
```

Slip 17_1 : Write a 'C' program to sort elements of a singly linked list in ascending order and display the sorted List.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
typedef struct node
{
	int data;
	struct node *next;
}NODE;
```

```
NODE *head=NULL;

void create()
{
        int i,n;
        NODE *t;
        printf("\nHow Many Nodes you Want to Create: ");
        scanf("%d",&n);

        head=(NODE *)malloc(sizeof(NODE));
        printf("\n Enter node 1: ");
        scanf("%d",&head->data);
        t=head;
        for(i=1;i<n;i++)
        {
                t->next=(NODE *)malloc(sizeof(NODE));
                t=t->next;
                printf("\n Enter node %d: ",i);
                scanf("%d",&t->data);
        }
        t->next=NULL;
}
void display()
{
        NODE *t;
        for(t=head;t!=NULL;t=t->next)
        {       printf("\t %d ->",t->data);  }
}
void sort()
{
        NODE *p,*q;
        int temp;
        for(p=head;p!=NULL;p=p->next)
        {
                for(q=p->next;q!=NULL;q=q->next)
                {       if(p->data > q->data)
                        {       temp = p->data;
                                p->data = q->data;
                                q->data = temp;
                        }
                }
        }
}
main()
{       int n;
         clrscr();
```

```
	create();
	printf("\n Link list is : ");
	display();
	printf("\n After sorting Link list is = ");
	sort();
	display();
	 getch();
}
```

Slip18_1 and Slip25_1 : sort data using Quick sort

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
int split(int a[],int lower,int upper)
{	int pivot,i,j,temp;
	i=lower+1;
	j=upper;
	pivot=a[lower];
	while(j>=i)
	{	while(pivot>a[i])
		i++;
		while(pivot<a[j])
			j--;
		if(i<j)
		{	temp=a[i];
			a[i]=a[j];
			a[j]=temp;
		}
	}
	temp=a[lower];
	a[lower]=a[j];
	a[j]=temp;
	return j;
}
void quicksort(int z[],int left,int right)
{	int i;
	if(right>left)
	{	i=split(z,left,right);
		quicksort(z,left,i-1);
		quicksort(z,i+1,right);
	}
}
void main()
{	int a[20],i,n;
		clrscr();
	printf("\n Enter how many elements would you like to enter : ");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{	printf("\n Enter data : ");
		scanf("%d",&a[i]);
	}
	quicksort(a,0,n-1);
	printf("\n The array after sorting is :\n");
	for(i=0;i<n;i++)
		printf("%d\t",a[i]);
		getch();
}
```

Slip19_1 : sort data using Merge sort

```
#include<stdlib.h>
#include<conio.h>
#include<stdio.h>
void merge(int a[10],int l,int m,int u)
{	int c[10],i,j,k;
	i=l;
	j=m+1;
	k=0;
	while(i<=m && j<=u)
	{	if(a[i]<a[j])
		{	c[k]=a[i];
			k++;
			i++;
		}
		else
		{	c[k]=a[j];
			k++;j++;
		}
	}
	while(i<=m)
	{	c[k]=a[i];
		i++;k++;
	}
```

```
        while(j<=u)
        {       c[k]=a[j];
                k++;j++;
        }
        for(i=l,j=0;i<=u;i++,j++)
                a[i]=c[j];
}
void merge_sort(int a[10],int i,int j)
{       int k=0;
        if(i<j)
        {       k=(i+j)/2;
                merge_sort(a,i,k);
                merge_sort(a,k+1,j);
                merge(a,i,k,j);
        }
}
void main()
{       int a[100],i,n;
         clrscr();
        printf("\n Enter how many elements would you like to enter:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {       printf("\n Enter data:");
                scanf("%d",&a[i]);
        }
        merge_sort(a,0,n-1);
        printf("\n The array after sorting is:\n");
        for(i=0;i<n;i++)
                printf("%d\t",a[i]);
}
```

Slip 20_1Write a 'C' program to read a postfix expression, evaluate it and display the result.

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#include<conio.h>
#define SIZE 20
int stk[SIZE];
int top;
void init()
{  top=-1;

}
int isfull()
{       if(top==SIZE-1)
```

```
            return 1;
            else
                        return 0;
}
int isEmpty()
{ if(top==-1)
            return 1;
            else
                        return 0;
}
void push(int c)
{           if(isfull())
            {        printf("Stack is overflow");    }
            else
            {     top++;
                        stk[top]=c;
            }
}
int pop()
{    if(isEmpty())
            {      printf("Stack is empty");  }
            else
            {   int c;
                        c=stk[top];
                        top--;
                        return c;
            }
}
void main()
{           int i,op1,op2,ans;
            char s[30];
            clrscr();
            printf("\nEnter the Postfix Expr");
            gets(s);
            init();
            for(i=0;s[i]!='\0';i++)
            {
                        if(s[i]<='9' && s[i]>='0')
                                    push(s[i]-48); push(atoi(s[i]));
                        else
                        {           op2=pop();
                                    op1=pop();
                                    switch(s[i])
                                    {
                                                case '+': push(op1+op2);
                                                            break;
                        case '-': push(op1-op2);
                                  break;
                        case '/': push(op1/op2);
                                  break;
                        case '*': push(op1*op2);
                                  break;
                        case '^': push(pow(op1^op2));
                                  break;
                        case '%':push(op1%op2);
                                 break;
                    }
                }
        } end of for
        ans=pop();
        printf("\n Result is %d ",ans);
         getch();

}
```

Slip 21_1create two singly linked lists and concatenate one list at the end of another list.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
        int data;
        struct node *next;
};

struct node* create(int n)
{        int i;
        struct node *f,*t;
        f=(struct node *)malloc(sizeof(struct node));
        printf("\n Enter node 1: ");
        scanf("%d",&f->data);
        t=f;
        for(i=1;i<n;i++)
        {
                t->next=(struct node *)malloc(sizeof(struct node));
                t=t->next;
                printf("\n Enter node %d: ",i);
                scanf("%d",&t->data);
        }        t->next=NULL;
        return f;
}
void display(struct node* f)
```

```
{           struct node *t;
            for(t=f;t!=NULL;t=t->next)
            {           printf("\t %d ->",t->data);
            }
}

struct node* concat(struct node* f1,struct node* f2)
{           struct node *t;
            for(t=f1;t->next!=NULL;t=t->next);
            t->next=f2;
            return f1;
}
void main()
{           int no,num;
            struct node *h1,*h2,*h3;
             clrscr();
            printf("\n Enter no of nodes for 1st link");
            scanf("%d",&no);
            h1=create(no);
            display(h1);
            printf("\n Enter no of nods for 2nd link");
            scanf("%d",&num);
            h2=create(num);
            printf("\n Concatination is ");
            h3=concat(h1,h2);
            display(h3);
                        getch();
}
```

Slip22_1 : create a singly link list and delete last node and insert it into first position

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
typedef struct node
{
            int data;
            struct node *next;
}NODE;
NODE *head=NULL;

void create()
{
            int i,n;
            NODE *t;
```

```
          printf("\nHow Many Nodes you Want to Create: ");
          scanf("%d",&n);

          head=(NODE *)malloc(sizeof(NODE));
          printf("\n Enter node 1: ");
          scanf("%d",&head->data);
          t=head;
          for(i=1;i<n;i++)
          {
                    t->next=(NODE *)malloc(sizeof(NODE));
                    t=t->next;
                    printf("\n Enter node %d: ",i);
                    scanf("%d",&t->data);
          }
          t->next=NULL;
}

void display()
{
          NODE *t;
          for(t=head;t!=NULL;t=t->next)
          {         printf("\t %d ->",t->data);  }
}
void delins()
{         int pos=0,i,no;
          struct node *t,*s,*nw;
          for(s=head;s!=NULL;s=s->next)
          {         pos++;
          }
          for(t=head,i=1;i<pos-1;t=t->next,i++);
          s=t->next;
          no=s->data;
          t->next=NULL;
                    free(s);
          nw=(NODE *)malloc(sizeof(NODE));
          nw->data=no;
          nw->next=head;
          head=nw;

}
void main()
{          clrscr();
          create();
          display();
          delins();
          printf("\n After performing operation \n");
```

```
            display();
             getch();
}
```

Slip23_1 : Write menu driven program using ‘C’ for Static implementation of Stack. The menu includes i)Insert ii) Delete iii)Display

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>
#define SIZE 40
int stk[SIZE];
int top;
void init()
{           top=-1;
}

int isfull()
{           if(top==SIZE-1)
            return 1;
            else
                        return 0;
}
int isempty()
{           if(top==-1)
            return 1;
            else
                        return 0;
}
void push(int c)
{           if(isfull())
            {           printf("Stack is overflow");
            }           else
            {            top++;
                        stk[top]=c;
            }
}
int pop()
{           if(isempty())
            {       printf("Stack is empty");
            }           else
            {           int c;
                        c=stk[top];
```

```
                top--;
                return c;
        }
}
void main()
{       int ch,i,n;
        init();
        do
        {   printf("\n 1.Insert \n 2.Delete \n 3.Display \n 4.Exit");
                printf("\n Enter choice ");
                scanf("%d",&ch);
                switch(ch)
                {   case 1: printf("\n Enter element ");
                        scanf("%d",&n);
                        push(n);
                        break;

                        case 2: printf("\n Deleted data is =%d",pop());
                                break;
                        case 3: for(i=top;i>=0;i--)
                                        printf("%d\t",stk[i]);
                                break;
                        case 4:break;
                }
        }       while(ch!=4);
}
```

Slip24_1 :count all non-zero elements, odd numbers and even numbers in the singly linked list.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
typedef struct node
{
        int data;
        struct node *next;
}NODE;
NODE *head=NULL;

void create()
{
        int i,n;
        NODE *t;
        printf("\nHow Many Nodes you Want to Create: ");
        scanf("%d",&n);
        head=(NODE *)malloc(sizeof(NODE));
        printf("\n Enter node 1: ");
        scanf("%d",&head->data);
        t=head;
        for(i=1;i<n;i++)
        {
                t->next=(NODE *)malloc(sizeof(NODE));
                t=t->next;
                printf("\n Enter node %d: ",i);
                scanf("%d",&t->data);
        }
        t->next=NULL;
}

void display()
{
        NODE *t;
        for(t=head;t!=NULL;t=t->next)
        {       printf("\t %d ->",t->data);  }
}

void count()
{       NODE *t;
        int nz=0,ec=0,oc=0;
        for(t=head;t!=NULL;t=t->next)
        {       if(t->data>0)
                nz++;
                if(t->data%2==0)
                        ec++;
                else
                        oc++;
        }

        printf("\n Non-zero elements =%d, \n odd numbers =%d \n even numbers
=%d",nz,ec,oc);
}
void main()
{ int n;
         clrscr();
        create();
        printf("\n Link list is : ");
        display();
        count();
          getch();
}
```

Slip26_1 : Circular Doubly link list

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
typedef struct node
{       int data;
        struct node *prev;
        struct node *next;
}NODE;
NODE *head=NULL;
void create()
{       NODE *t;
        int i,no;
        printf("enter no of nodes");
        scanf("%d",&no);
        head=(struct node*)malloc(sizeof(struct node));
        printf("\n Enter data");
        scanf("%d",&head->data);
        t=head;
        for(i=2;i<=no;i++)
        {       t->next=(struct node*)malloc(sizeof(struct node));
                t->next->prev=t;
                t=t->next;
                printf("\n Enter data");
                scanf("%d",&t->data);
                t->next=NULL;
        }
        t->next=head;
        head->prev=t;
}
void display()
{       NODE *t;
        t=head;
        do
        {
                printf("%d\t",t->data);
                t=t->next;
        }while(t!=head);
}
void main()
{
         clrscr();

        create();
        display();
```

```
            getch();
}
```

Slip27_1 :swap mth and nth element of singly linked list.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
typedef struct node
{
        int data;
        struct node *next;
}NODE;
NODE *head=NULL;

void create()
{
        int i,n;
        NODE *t;
        printf("\nHow Many Nodes you Want to Create: ");
        scanf("%d",&n);

        head=(NODE *)malloc(sizeof(NODE));
        printf("\n Enter node 1: ");
        scanf("%d",&head->data);
        t=head;
        for(i=1;i<n;i++)
        {
                t->next=(NODE *)malloc(sizeof(NODE));
                t=t->next;
                printf("\n Enter node %d: ",i);
                scanf("%d",&t->data);
        }
        t->next=NULL;
}

void display()
{
        NODE *t;
        for(t=head;t!=NULL;t=t->next)
        {       printf("\t %d ->",t->data);  }
}

void swap()
```

```
{
	int m,n,i,tmp;
	struct node *q,*ptr,*ptr1;
	printf("\nEnter the mth and nth position");
	scanf("%d%d",&m,&n);
	for(i=1,ptr=head;i<m && ptr!=NULL;i++);
	ptr=ptr->next;
	for(i=1,ptr1=head;i<n && ptr1!=NULL;ptr1=ptr1->next,i++);
	if(ptr!=NULL && ptr1!=NULL)
	{ tmp=ptr->data;
		ptr->data=ptr1->data;
		ptr1->data=tmp;
	}
	else
	{  printf("\nPosition Not Found");
	}
}

void main()
{	int no,num;
	struct node *h,*h1;
		clrscr();
	create();
	display();
	swap();
	display();
	getch();
}
```

Slip28_1 : Doubly link list

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
typedef struct node
{	int data;
	struct node *prev;
	struct node *next;
}NODE;
NODE *head=NULL;
void create()
{	NODE *t;
	int i,no;
	printf("enter no of nodes");
	scanf("%d",&no);
	head=(struct node*)malloc(sizeof(struct node));
	printf("\n Enter data");
	scanf("%d",&head->data);
	head->prev=NULL;
	t=head;
	for(i=2;i<=no;i++)
	{	t->next=(struct node*)malloc(sizeof(struct node));
		t->next->prev=t;
		t=t->next;
		printf("\n Enter data");
		scanf("%d",&t->data);
		t->next=NULL;
	}
	t->next=NULL;
}
void display()
{	NODE *t;
	for(t=head;t!=NULL;t=t->next)
		printf("%d\t",t->data);
}
void main()
{
	 clrscr();

	create();
	display();
	 getch();
}
```

Slip 29_1 :Write a ‘C’ program to create to a Singly linked list. Accept the number from user, search the number in the list .If the number is present delete the node from the list and add it at the beginning and display the list .If node not present print the message “Node not Found”.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
typedef struct node
{
	int data;
	struct node *next;
}NODE;
NODE *head=NULL;
```

```
void create()
{
	int i,n;
	NODE *t;
	printf("\nHow Many Nodes you Want to Create: ");
	scanf("%d",&n);

	head=(NODE *)malloc(sizeof(NODE));
	printf("\n Enter node 1: ");
	scanf("%d",&head->data);
	t=head;
	for(i=1;i<n;i++)
	{
		t->next=(NODE *)malloc(sizeof(NODE));
		t=t->next;
		printf("\n Enter node %d: ",i);
		scanf("%d",&t->data);
	}
	t->next=NULL;
}

void display()
{
	NODE *t;
	for(t=head;t!=NULL;t=t->next)
	{	printf("\t %d ->",t->data);  }
}

void delete()
{
	struct node *s,*temp,*nw;
	int cnt=0,i,sr,no,p=0,flag=0;
	printf("\n Node to be Search : ");
	scanf("%d",&sr);
	for(s=head;s!=NULL;s=s->next)
	{cnt++;}
	for(s=head;s!=NULL;s=s->next)
	{	p++;
		if(sr==s->data)
		{flag=1;
			break;}
	}
	if(flag==0)
	{
		printf("Data Not Found = %d ",sr);
```

```
        }
        else
        {
                if(p==1)
                {}
                else
                {
                        for(i=1,s=head;i<p-1;i++,s=s->next);
                        temp=s->next;
                        no=temp->data;
                        s->next=temp->next;
                        nw=(NODE *)malloc(sizeof(NODE));
                        nw->data=no;
                        nw->next=head;
                        head=nw;
                }
        }
        display();
        /*while(cnt!=0)
          { for(t=head,i=1;i<=cnt-1;i++)
          {  t=t->next;  }
          printf("%d ->\t",t->data);
          cnt--;
          }*/
}
void main()
{ int n;
         clrscr();
        create();
        printf("\n Link list is : ");
        display();
        delete();
          getch();
}
```

Slip30_1 : accept the details of employees from user and display it on the screen using Dynamic Memory Allocation.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
typedef struct node
{       int eno;
        char ename[20],eaddr[20];
        struct node *next;
}NODE;
NODE *head=NULL;

void create()
{       NODE *t;
        int i,no;
        printf("enter total no of employees");
        scanf("%d",&no);

        head=(NODE *)malloc(sizeof(NODE));
        printf("\n Enter eno, ename and eaddr");
        scanf("%d %s %s",&head->eno,head->ename,head->eaddr);
        t=head;
        for(i=2;i<=no;i++)
        {       t->next=(NODE*)malloc(sizeof(NODE));
                t=t->next;
                printf("\n Enter eno, ename and eaddr");
                scanf("%d %s %s",&t->eno,t->ename,t->eaddr);
        }
        t->next=NULL;
}
void display()
{       NODE *t;
        for(t=head;t!=NULL;t=t->next)
        {
                printf("\n");
                printf("%d\t%s\t%s",t->eno,t->ename,t->eaddr);
        }
}
void main()
{
         clrscr();
        create();
        display();
         getch();
}
```