**ASSIGNMENT 1**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct set

{ int a;

struct set \* node;

}\*c[2],\*temp,\*u;

void display(int r)

{

struct set \* p=c[r];

while(p!=NULL)

{printf("%d, ",p->a);

p=p->node; }

}

void skew(int i,int j)

{ struct set \*p=c[i];

struct set \*q=c[j];

while(p!=NULL)

{

while(q!=NULL)

{

if(p->a==q->a)

goto ab;

q=q->node;}

printf("%d ,",p->a);

ab:

p=p->node;}

}

void intersection()

{ struct set \*p=c[0];

struct set \*q=c[1];

while(p!=NULL)

{

while(q!=NULL)

{

if(p->a==q->a)

printf("%d",p->a);

q=q->node;}

p=p->node;

}

}

int create(int r)

{ int k;

struct set \*y;

temp = (struct set\*)malloc(sizeof(struct set));

printf("enter the data - ");

scanf("%d",&k);

y=c[r];

while(y!=NULL)

{

if(y->a==k)

{return 0;}

y=y->node;

}

temp->a=k;

temp->node=NULL;

if(c[r]==NULL)

c[r]=temp;

else

{

u=c[r];

c[r]=temp;

temp->node=u;

}

return 0;

}

void main()

{

int a;

ab: clrscr();

printf("..::MENU::..\n\n1.NEW ENTRY IN SET 1 \n2.NEW ENTRY IN SET2");

printf("\n\n\n\n3.UNION\n4.INTERSECTION\n5.DIFFERENCE\n6.Skew-Difference\n\t\t");

scanf("%d",&a);

printf("{ ");

switch(a)

{ case 1:

create(0);

goto ab;

case 2:

create(1);

goto ab;

case 3:

display(0);

display(1);

goto ab;

case 4:

intersection();

goto ab;

case 5:

skew(0,1);

goto ab;

case 6:

skew(0,1);

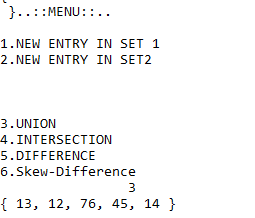
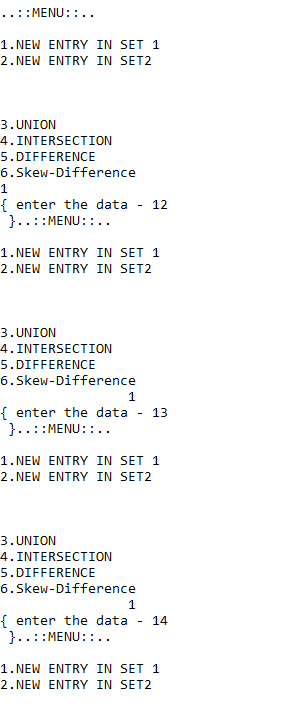
skew(1,0);

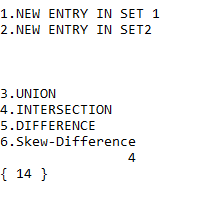
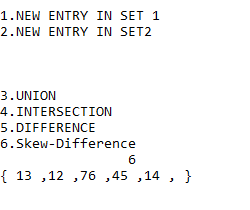
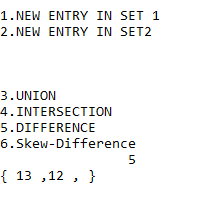
goto ab;

default:

printf(" PROGRAM TERMINATED }");}

getch();}





**ASSIGNMENT 2**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void addition(int \*\*a,int \*\*b,int c,int d,int e,int f)

{

int i,j;

if(c==e&&d==f)

{

for( i=0;i<c;i++)

{ printf("\n");

for( j=0;j<d;j++)

{ printf(" %d",\*(a[i]+j)+\*(b[i]+j));}}

}

else

printf("..::SORRY! NO ADDITION POSSIBLE ::..");

}

int multi(int \*\*a,int \*\*b,int c,int d,int e,int f)

{ int i,j,temp,k,abg=0;

if(e!=d)

{printf("SORRY! no multiplication possible : ");}

for(i=0;i<c;i++)

{ printf("\n");

for(j=0;j<d;j++)

{ abg=0;

for(k=0;k<f;k++)

{ abg=abg+ ((\*(a[i]+k)) \* (\*(b[k]+j)));}

printf("%d ",abg);}}

return 0;

}

int \*\* create(int a,int g)

{ int \*\*b,i,q,p;

b=(int \*\* )malloc(a\*sizeof(int\*));

for( i=0;i<g;i++)

b[i]=(int \*)malloc(g\*sizeof(int));

printf("NOW ENTER THE DATA - \n\t\t");

for( q=0;q<a;q++)

{ for( p=0;p<g;p++)

{ scanf("%d",b[q]+p );}}

return b;}

void transpose(int \*\*a,int d,int e)

{ int i,j,temp;

for( i=0;i<d;i++)

{ for(j=i;j<e;j++)

{a[i][j]^=a[j][i]^=a[i][j]^=a[i][j]^=a[i][j];}}

for( i=0;i<d;i++)

{ printf("\n");

for(j=0;j<e;j++)

{ printf("%d ",\*(a[i]+j));}}

}

int smallest (int \*p,int e)

{ int k=p[0],l=0,i;

for( i=0;i<e;i++)

if(k>p[i])

{k=p[i]; l=i;}

return l;

}

void largest( int \*\*a,int d,int e)

{ int large=a[0][e],i;

for( i=0;i<d;i++)

if(a[i][e]>large)

large=a[i][e];

printf("%d",large);

}

int saddle(int \*\*a,int d,int e)

{ int \*p,r,i;

p=a[0];

r = smallest(p,e);

for(i=1;i<d;i++)

{ p=a[i];

if(r!=smallest(p,e))

{ printf(" saddle point not exist ") ;

return 0;}}

largest(a,d,r);

return 0;

}

void main()

{ char a;

int \*\*b,\*\*c,f,g,d,e;

printf("MENU\n\n\n1. PRESS 1 for ADDITION\n2. PRESS for MULTIPLICATION\n3. TRANSPOSE\n4. SADDLE");

printf("\n\n\n\t\tNOW ENTER YOUR CHOICE : ");

a=getch();

printf("enter the size of first matrix ");

scanf("%d%d",&d,&e);

b=create(d,e);

if(a<'3')

{ printf("enter the size of second matrix ");

scanf("%d%d",&f,&g);

c=create(f,g);}

switch(a)

{ case '1':

addition(b,c,d,e,f,g);

break;

case '2':

multi(b,c,d,e,f,g);

break;

case'3':

transpose(b,d,e);

break;

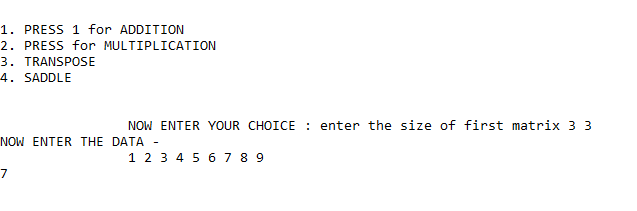
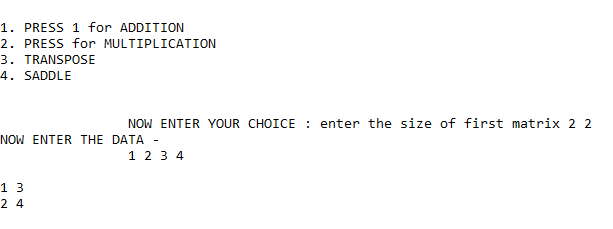
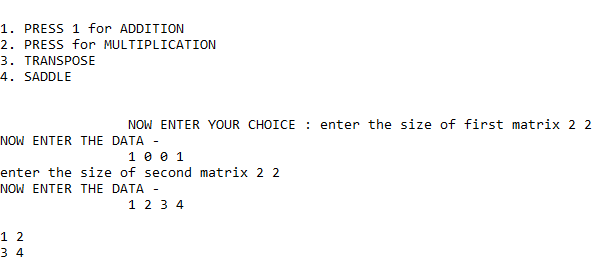
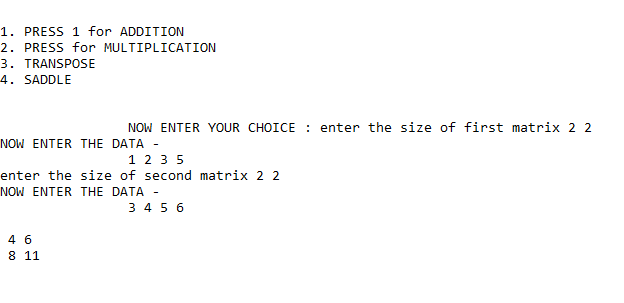
case '4':

saddle(b,d,e);

}

getch();

}

****

**ASSIGNMENT 3**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void reverse(char \*a)

{ int u strlen(a),i;char g;

printf("\n%s IS CURRENT STRING ",a);

for(i=0;i<(u/2);i++)

{a[i]^= a[u-i-1]^=a[i]^= a[u-i-1];}

printf(" \n\nNOW! IT CHANGES TO %s",a);

}

void copy(char \*a,char \*b)

{ int u= strlen(a),i; char g;

printf("\n%s IS CURRENT STRING 1 ",a);

printf("\n%s IS CURRENT STRING 2 CONTAIN GARBAGE \n\n\n\n",b);

for( i=0;i<u;i++)

b[i]=a[i];

b[u]='\0';

printf("\n%s IS new STRING 1 ",a);

printf("\n%s IS new STRING 2 WHICH IS COPIED VALUE\n\n",b);

}

int compare(char c[],char d[])

{ int i= strlen(c),j= strlen(d),k,l;

if(i>=j)

k=i;

else

k=j;

for(l=0;l<k;l++)

{ if(c[l]!=d[l])

{ if(c[l]>d[l])

{ printf("\n\n\t STRING 1 IS GREATER ");}

else

printf("\n\n\t STRING 2 IS GREATER");

return 0;}}

printf("\n\n\tboth strings are same ");

return 0;

}

int palindrome(char \*a)

{ int u,i;char g;

printf("\n%s IS CURRENT STRING ",a);

for(i=0;i<u/2;i++)

{ if(a[i]!=a[u-i-1])

{printf("\n\t and it is NOT A PALINDROME");

return 0;}}

printf("\n\t and it is A PALINDROME");

return 0;

}

void search(char \*c,char e)

{ int t,i,r=0;

printf("\n\nSTRING 1 CONTAINS %s \n",c);

printf("\n%c TO BE SEARCHED \n\n\n",e);

t=strlen(c);

for(i=0;i<t;i++)

if(c[i]==e)

printf("\n\tSTRING FOUND AT POSITION %d\n",i+1);

if(r==0)

printf("\n\tSTRING NOT FOUND");

}

void main()

{ char a,\*d,\*c,e; xy:

printf("MENU\n\n\n1.REVERSE\n2.PALINDROME\n3.COPY\n4.COMPARE \n5.SEARCH");

printf("\nclick other number to TERMINATE...!!!\n\n\t\t\t");

a=getch();

switch(a)

{ case '1':

reverse(c);

goto xy;

case '2':

palindrome(c); getch();

goto xy;

case '3':

copy(c,d); getch();

goto xy;

case '4':

compare(c,d); getch();

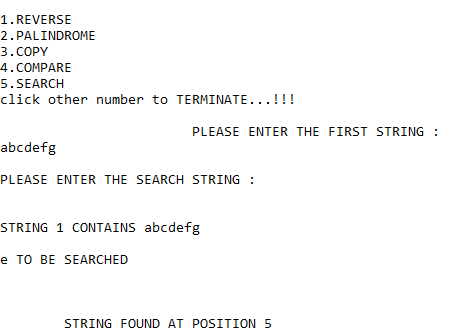
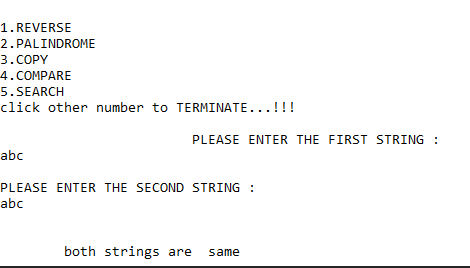
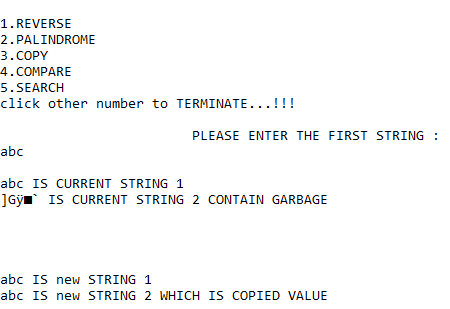
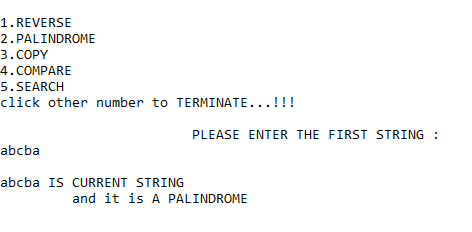
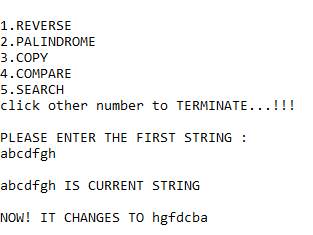
goto xy;

case '5':

search(c,e); getch();

goto xy;}

}



**ASSIGNMENT 4**

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

struct employee

{ int t;

int code;

char name[100];

union asd

{ int salary;

char grade;

}an;

} emo[50];

struct employee newemp()

{ int a;

struct employee w;

printf("\n\n\nENTER THE RECORD OF EMPLOYEE \n");

printf("\n\nENTER THE EMPLOYEE CODE ");

scanf("%d",&w.code);

printf("\nENTER THE EMPLOYEE NAME ");

scanf("%s",&w.name);

printf("ENTER THE TYPE YOU WANT TO INSERT ");

scanf("%d",&a);

w.t=a;

if(a==1)

{printf("\nENTER THE EMPLOYEE salary ");

scanf("%d",&w.an.salary);}

if(a==2)

{printf("\n\nENTER THE GRADE ");

scanf("%s",&w.an.grade);}

return w;}

void deleteer(int i,int t)

{ int y;

for(y=i;y<t;y++)

{ emo[y]=emo[y+1];}}

void output(int i)

{ printf("\n\n EMPLOYEE CODE ");

printf("%d",emo[i].code);

printf("\nENTER THE EMPLOYEE NAME ");

printf("%s",emo[i].name);

if(emo[i].t==1)

{ printf("\nENTER THE EMPLOYEE salary ");

printf("%d",emo[i].an.salary);}

if(emo[i].t==2)

{ printf("\nENTER THE GRADE ");

printf("%c",emo[i].an.grade);}

}

int search(int r,int y,int t)

{ int i;

struct employee f;

if(y==0)

for( i=0;i<50;i++)

if(emo[i].code==r)

{ output(i);

return 0;}

if(y==1)

{ for( i=0;i<50;i++)

if(emo[i].code==r)

{ f=newemp();

emo[i]=f;

return 0;}}

if(y==2)

{ for( i=0;i<50;i++)

if(emo[i].code==r)

{deleteer(i,t);return 0; }}

return 0;

}

int main()

{ int t=0,I; char a;int r;struct employee nd;ab:

printf("\n\n\n\t\t\t\t..::MENU::..\n\n");

printf("PRESS 1. NEW EMPLOYEE RECORD ENTRY\nPRESS 2. DISPLAY\nPRESS 3. DISPLAY ALL\nPRESS 4. MODIFY PARTICULAR\nPRESS 5. DELETE\n\n\n\t\t\t");

a=getch();

switch(a)

{ case '1':

nd=newemp();

emo[t]=nd;++t;

goto ab;

case '2':

printf("enter the employee code to be displayed : ");

scanf("%d",&r);

r=search(r,0,t);

if(r>50)

printf("NO SUCH DATA PRESENT ");goto ab;

case '3':

for( i=0;i<t;i++)

{output(i);}

goto ab;

case '4':

printf("enter the employee code to be modified : ");

scanf("%d",&r);

r=search(r,1,t);

if(r>50)

printf("NO SUCH DATA PRESENT ");

goto ab;

case '5':

printf("enter the employee code to be deleted : ");

scanf("%d",&r);

r=search(r,2,t);

if(r>50)

printf("NO SUCH DATA PRESENT ");

else

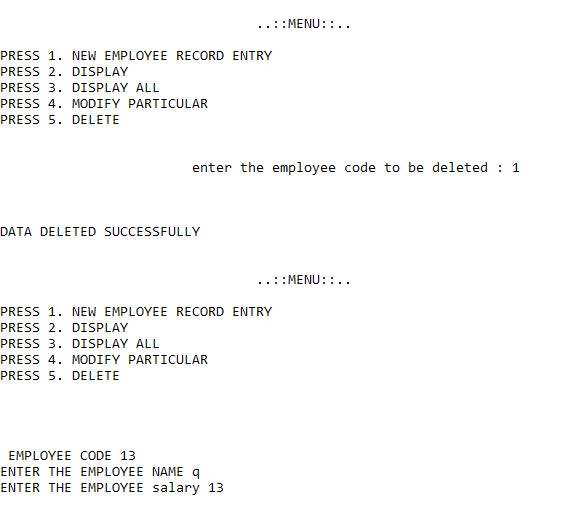
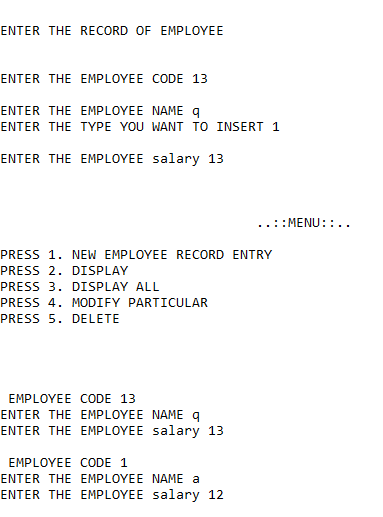
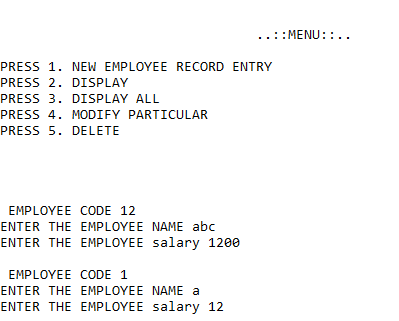
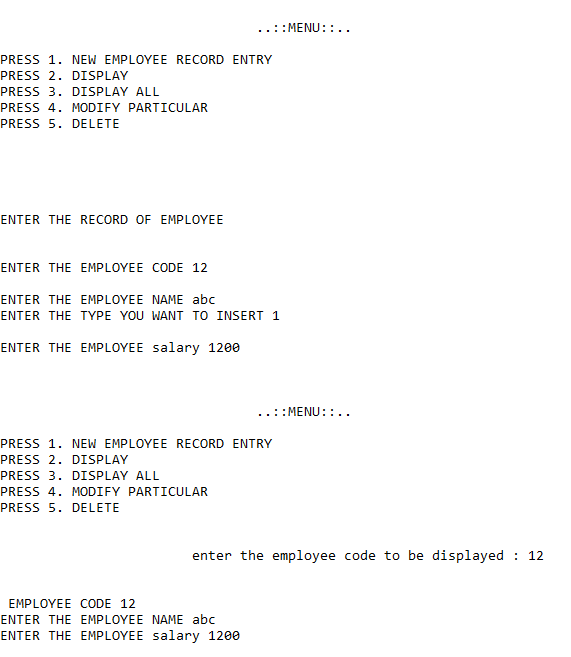
{ printf("\n\n\nDATA DELETED SUCCESSFULLY ");--t;}

goto ab;

}

return 0;

}



**ASSIGNMENT 5**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void BUBBLE(char u[],int r)

{ int i,j; char temp;

printf("\n\n\n BUBBLE SORT OF STRING %s",u);

for( i=0;i<r;i++)

{ for( j=0;j<r-1-i;j++)

{ if(u[j]>u[j+1])

{ u[j]^=u[j+1]^=u[j]^=u[j+1]; }}

printf("\n %s",u); }

}

void SELECTION(char \*t,int r)

{ char larg,temp; int y,i,j;

for( i=0;i<r-1;i++)

{ larg=t[i];y=i;

for( j=i+1;j<r;j++)

{ if(t[j]>larg)

{ larg=t[j]; y=j; }}

t[i]^=t[y]^=t[i]^=t[y];

printf("\n\n%s",t); }

}

int Bsearch(char \*t,int start,int last,char u)

{ int mid=(start+last)/2;

printf(" %d %d ",start,last);

if(last<start)

{ printf(" NOT FOUND STRING IS NOT SORTED ");

return 0;}

if(start==last)

{ if(t[mid]==u)

{printf(" FOUND ");

return 0;}

if(t[mid]!=u)

{printf(" SORRY NOT FOUND ");

return 0;} }

else

{ if(t[mid]==u)

{ printf("\n\n\n FOUND ");

return 0; }

if(t[mid]<u)

{start=mid+1; Bsearch(t,start,last,u);}

if(t[mid]>u)

{last=mid-1; Bsearch(t,start,last,u);}}}

void bsearch(char \*t,int r)

{ int start,ends,mid; char a;

printf("\n\n\nENTER THE CHARECTER TO BE SEARCHED - ");

a=getch();

start=0; ends=r-1; mid=(start+ends)/2;

while(start<=ends)

{ if(t[mid]==a)

{ printf("\n\n FOUND at %d",mid);

break; }

if(t[mid]<a)

{ start=mid+1;}

if(t[mid]>a)

{ ends=mid-1;}

mid=(start+ends)/2;}

}

void INSERTION(char \*t, int r)

{ int i,temp,j;

for( i=1;i<r;i++)

{ temp=t[i];

j=i-1;

while(temp<t[j])

{ t[j+1]=t[j]; --j; }

t[j+1]=temp;

printf("\n\n%s",t); }}

int main()

{ int p; char a,c; char \*w=(char \*) malloc (sizeof(char)\*50);ab:

printf("\n\n\n\t\t\t ENTER THE STRING TO PERFORM FURTHER OPERATIONS ");

scanf("%s",w);

printf("\n\n1.BUBBLE SORT\n2.SELECTION SORT\n3.INSERTION SORT\n4.BINARY SEARCH\n5.RECURSIVE BINARY SEARCH");

a=getch();

switch(a)

{ case '1':

BUBBLE(w,p);

case '2':

SELECTION(w,p);

case '3':

INSERTION(w,p);

case '4':

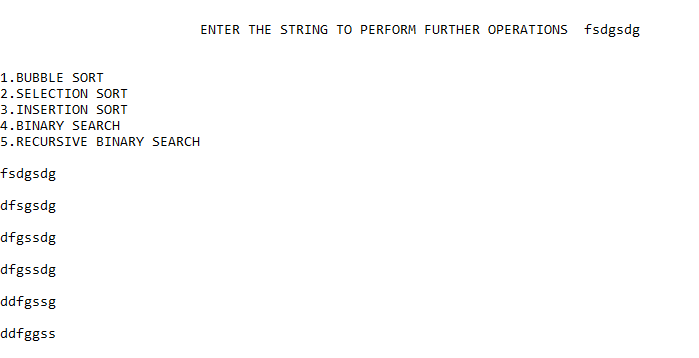
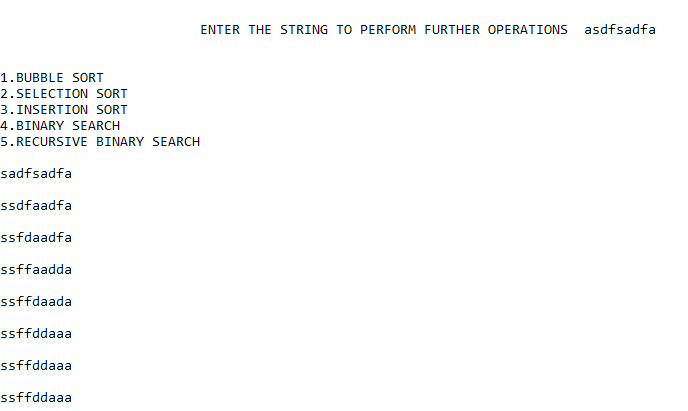
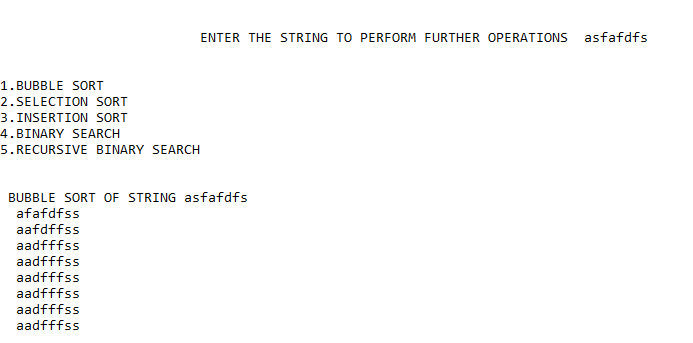
bsearch(w,p);

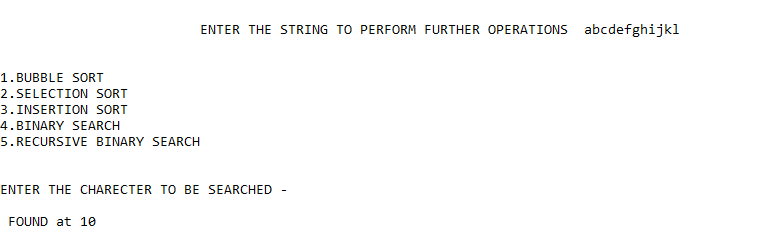
case '5':

printf(" \n\n ENTER THE CHARECTER TO BE SEARCHED ");

c=getch();

Bsearch(w,0,p-1,c); }}





**ASSIGNMENT 6**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct student

{ char name[50];

int rno;

int classs; };

struct student \* getdata()

{ struct student \* a=(struct student\*)malloc(sizeof(struct student));

printf("\n\n\nenter the name of student - ");

scanf("%s",a->name);

printf("\nenter the roll number ");

scanf("%d",&a->rno);

printf("\nenter the class ");

scanf("%d",&a->classs);

return a;

}

int display(int y)

{ int f; struct student q; FILE \*a =fopen("abc.dat","r");

if(y==0)

{ printf("\n\nENTER THE ROLL NUMBER TO BE DISPLAYED ");

scanf("%d",&f); }

printf("\n\n\n STUDENT's NAME\t\t\tROLL NUMBER\t\t\tCLASS");

while(!feof(a))

{ fscanf(a,"%s%d%d",&q.name,&q.rno,&q.classs);

if(q.rno==f&&y==0)

{ printf("\n\n%s\t\t\t\t%d\t\t\t%d",q.name,q.rno,q.classs);

fclose(a);

return 1; }

if(y==1)

printf("\n\n%s\t\t\t\t%d\t\t\t%d",q.name,q.rno,q.classs);}

fclose(a);

printf("\n\n\n\n\n");

return 0;

}

void deletes()

{ int as; FILE \*a,\*b; struct student w;

a=fopen("abc.dat","a+");

b=fopen("abcd.dat","w");

printf("\n\n\nENTER THE ROLL NUMBER TO BE DELETED ");

scanf("%d",&as);

while(!feof(a))

{ fscanf(a,"%s%d%d",&w.name,&w.rno,&w.classs);

if(as!=w.rno)

fprintf(b,"%s %d %d",w.name,w.rno,w.classs); }

fclose(a);

fclose(b);

remove("abc.dat");

rename("abcd.dat","abc.dat");

}

int modify()

{ FILE \* a; int qa,t; struct student q; struct student \*abh; int y;

a=fopen("abc.dat","r+");

printf("\n\n\nenter the ROLL NO to be modified - ");

scanf("%d",&qa);

while(!feof(a))

{ t=ftell(a);

fscanf(a,"%s%d%d",&q.name,&q.rno,&q.classs);

if(qa==q.rno)

{ fseek(a,t,0);

abh=getdata();

y=searches(abh->rno);

if(y)

{ printf(" \n\nDATA NOT CREATED \n\n"); return 0;}

fprintf(a,"%s %d %d",abh->name,abh->rno,abh->classs);

printf("\n\nDATA INSERTED SUCCESSFULLY \n\n");

fclose(a);

return 0;} }

return 0;

}

int searches(int t)

{ FILE \*a;

struct student q;

a=fopen("abc.dat","r");

while(!feof(a))

{ fscanf(a,"%s%d%d",&q.name,&q.rno,&q.classs);

if(q.rno==t)

{ close(a);

return 1;}}

fclose(a);

return 0; }

int create()

{ struct student \*abh; FILE \*a; int y;

a=fopen("abc.dat","a+");

abh=getdata();

y=searches(abh->rno);

if(y) printf(" \n\nDATA NOT CREATED \n\n");

fprintf(a,"%s %d %d",abh->name,abh->rno,abh->classs);

printf("\n\nDATA INSERTED SUCCESSFULLY \n");

fclose(a);

return 1;

}

int main()

{

char a;int t;

ab:

printf("\n\n\n\t\t\t\t..:: MENU ::..\n\n");

printf("1.INSERT\n2.MODIFY\n3.DELETE\n4.DISPLAY\n5.DISPLAY ALL\n\n\n\t\t\t");

a=getch();

switch(a)

{

case'1':

create();

goto ab;

case '2':

modify();

goto ab;

case '3':

deletes();

goto ab;

case '4':

t=display(0);

if(!t)

{printf("\n\n .. SORRY..!! NO SUCH DATA FOUND \n\n");}

goto ab;

case '5':

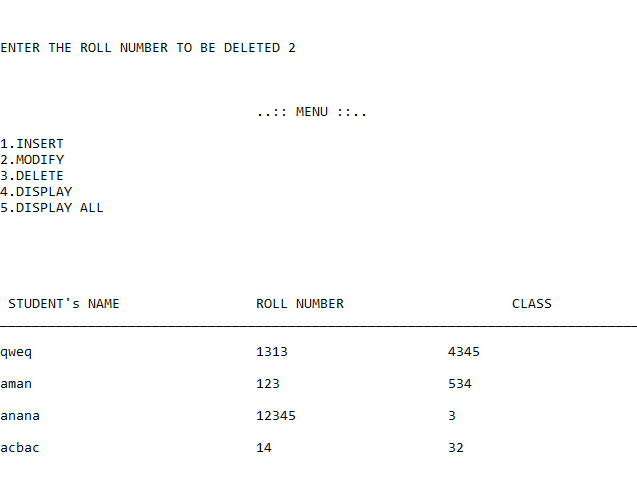
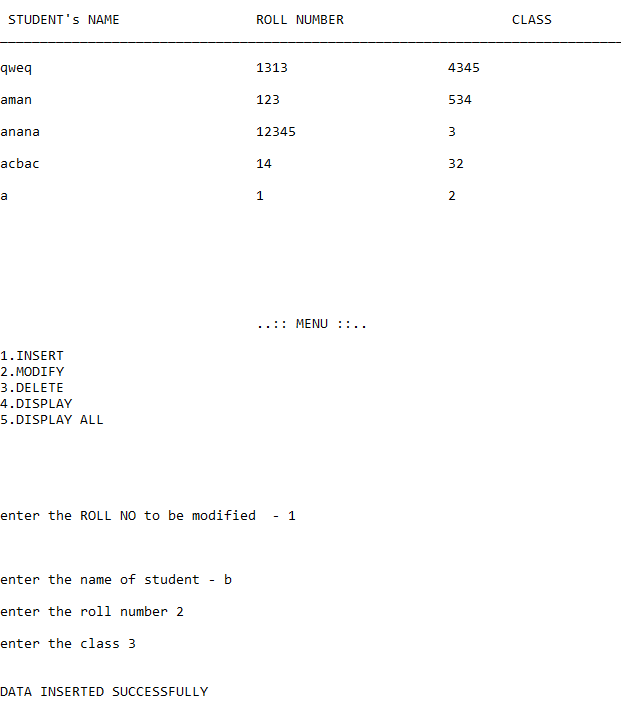
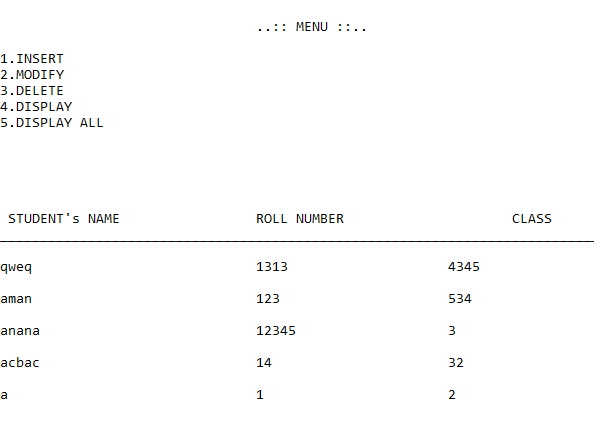
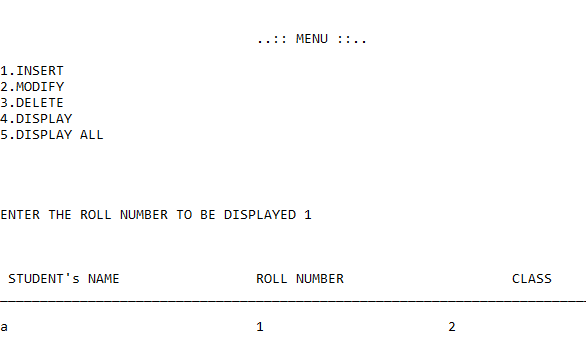
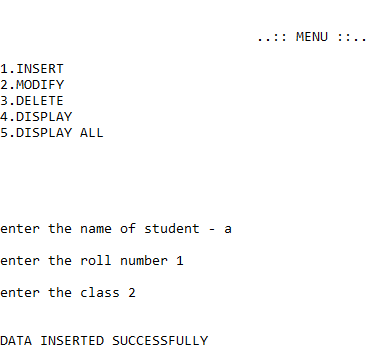
display(1);

goto ab;

}

return 0;

}



**ASSIGNMENT 7**

#include<stdio.h>

#include<stdlib.h>

int \* sorts(int \*a,int l,int h)

{ int mid,\*b,\*c,r,k,i,p,s;

if(l<h)

{ mid=((l+h)/2);

r=(mid-l)+1;

k=(h-mid);

b=(int\*)malloc(sizeof(int)\*r);

c=(int\*)malloc(sizeof(int)\*k);

for(i=0;i<r;i++)

{b[i]=a[i];

printf(" %d ",b[i]);}

printf("\n");

for(i=0;i<k;i++)

{ \*(c+i)=a[i+mid+1];

printf(" %d ",\*(c+i)); }

printf("\n\n\n\n\n");

b=sorts(b,l,mid);

c=sorts(c,0,k-1); p=0;s=0;

for(i=0;i<=h;i++)

{ if(p==r)

{b[p]=c[s]+12;}

if(s==k)

c[s]=b[p]+12;

if(b[p]<=c[s])

{a[i]=b[p]; ++p;}

else

{a[i]=c[s];

++s; }}

return a; }

return a; }

int main()

{ int \*a,q,i;

scanf("%d",&q);

a=(int \*)malloc(sizeof(int)\*q);

for(i=0;i<q;i++)

scanf("%d",&a[i]);

a=sorts(a,0,q-1);

printf("\n\n\n\n");

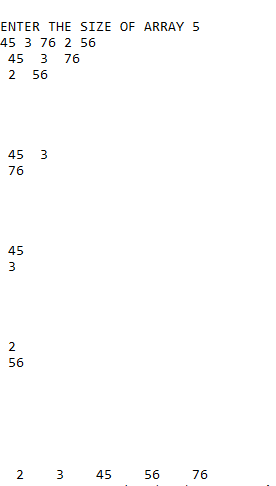
for(i=0;i<q;i++)

printf(" %d ",a[i]);

free(a);

return 0;

}



**ASSIGNMENT 8**

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

int\*\* sparse(int b,int c,int \*ja)

{ int \*\*a,i=0,j=0,s=0,\*\*r,sa=0;

\*ja=0;

a=(int\*\*)malloc(b\*sizeof(int\*));

for(i=0;i<b;i++)

for(j=0;j<b;j++)

a[i]=(int\*)malloc(c\*sizeof(int));

for(i=0;i<b;i++)

for(j=0;j<c;j++)

{ scanf("%d",&a[i][j]);

if(a[i][j]!=0)

++\*ja; }

r=(int\*\*)malloc(sizeof(int\*)\*3);

r[0]=(int\*)malloc(sizeof(int)\*\*ja);

r[1]=(int\*)malloc(sizeof(int)\*\*ja);

r[2]=(int\*)malloc(sizeof(int)\*\*ja);

for(i=0;i<b;i++)

for(j=0;j<c;j++)

if(a[i][j]!=0)

{r[0][sa]=i;

r[1][sa]=j;

r[2][sa]=a[i][j];

++sa; }

return r;

}

int \*\* transpose(int \*\*l, int \*ja, int c)

{ int i,\*q,j,temp;

if(c==1)

{goto ab;}

l[0]l^=[1]l^=[0]l^=[1];

ab:

for(j=0;j<\*ja-1;j++)

for(i=0;i<\*ja-1;i++)

if((l[0][i]\*10 + l[1][i]) > (l[0][i+1]\*10 + l[1][i+1]) )

{

temp=l[0][i];

l[0][i]=l[0][i+1];

l[0][i+1]=temp;

temp=l[1][i];

l[1][i]=l[1][i+1];

l[1][i+1]=temp;

temp=l[2][i];

l[2][i]=l[2][i+1];

l[2][i+1]=temp; }

return l; }

int\*\* mul(int \*\*l,int\*\*m,int\*ja,int\*na,int b,int c,int d,int e)

{ int \*\*q,\*p,\*r, i,j,k=0,z=0,x=0,temp=0,s=0,a=0;

q=(int\*\*)malloc((sizeof(int\*)\*3));

m[1^]=m[0]^=m[1]^=m[0];

for( i=0;i<3;i++)

q[i]=(int\*)malloc(sizeof(int)\*0); /\*0 byte array created \*/

for(i=0;i<b;i++)

{ p=(int\*)calloc(c,sizeof(int));

r=(int\*)calloc(c,sizeof(int));

while(l[0][z]==i)

{ p[l[1][z]]=l[2][z]; ++z; }

x=0;

for(j=0;j<e;j++)

while(m[1][x]==j)

{ r[m[0][x]]=m[2][x]; ++x; }

temp=0;

for(k=0;k<c;k++)

temp=temp+(p[k]\*r[k]);

if(temp!=0)

{ a=a+1;

for(s=0;s<3;s++)

\*(q+s)=(int\*)realloc(\*(q+s),sizeof(int)\*(a));

q[0][a-1]=j;

q[1][a-1]=i;

q[2][a-1]=temp;}} }

transpose(q, &a, 1);

for(i=0;i<a;i++)

printf("%d %d %d \n",q[0][i],q[1][i],q[2][i]);

return q;

}

int \*\* add(int \*\*l, int\*\* m, int \*ja,int \*ka)

{ int i=0,j=0,c=0,temp;

for(i=0;i<\*ja;)

{ printf("%d %d %d \n\n",l[0][i],l[1][i],l[2][i]);

for(j=0;j<=\*ka;j++)

if((l[0][i]\*10 +l[1][i])==(m[0][j]\*10 +m[1][j]))

m[2][i]=l[2][i]+m[2][j];

goto ab; }

++ \*ka;

for(c=0;c<3;c++)

{ m[c]=(int \*)realloc(m[c],sizeof(int)\*( \*ka));}

m[0][(\*ka)-1]=l[0][i];

m[1][(\*ka)-1]=l[1][i];

m[2][(\*ka)-1]=l[2][i];

ab: ++i;

}

for(j=0;j<\*ka-1;j++)

for(i=0;i<\*ka-1;i++)

{ if((m[0][i]\*10 + m[1][i]) > (m[0][i+1]\*10 + m[1][i+1]) )

{ temp=m[0][i];

m[0][i]=m[0][i+1];

m[0][i+1]=temp;

temp=m[1][i];

m[1][i]=m[1][i+1];

m[1][i+1]=temp;

temp=m[2][i];

m[2][i]=m[2][i+1];

m[2][i+1]=temp; }

for(i=0;i<\*ka;i++)

printf("%d %d %d\n",m[0][i],m[1][i],m[2][i]);

return m;

}

int main()

{ int b,c,\*\*l,i,j,\*ja=(int\*)malloc(sizeof(int)),\*\*m,\*na,d,e,\*\*p; char q; ab:

scanf("%d%d",&b,&c);

l=sparse(b,c,ja);

if(q!='1')

{ na=(int\*)malloc(sizeof(int));

scanf("%d%d",&d,&e);

m=sparse(d,e,na); }

switch(q)

{ case'1':

transpose(l,ja,0);

for(i=0;i<\*ja;i++)

printf("\n%d %d %d ",l[0][i],l[1][i],l[2][i]);

goto ab;

case'2':

if(b!=d || c!=e)

{printf("\n\n\n\t\t\t..:: NOT POSSIBLE ::..\n\n"); goto ab;}

else

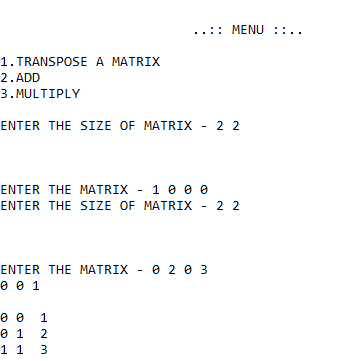
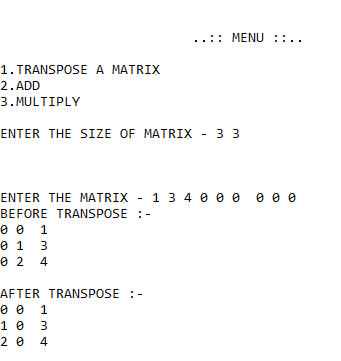
{p=add(l,m,ja,na); goto ab;}

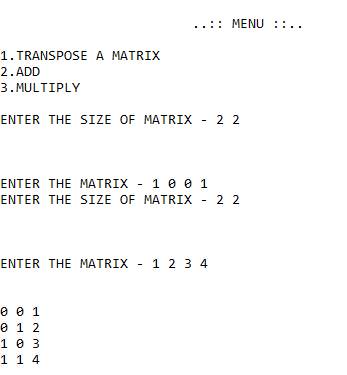
case'3':

if(c!=d) {printf(" NO MULTIPLICATION POSSIBLE ");goto ab;}

mul(l,m,ja,na,b,c,d,e); goto ab; }

return 0;}





**ASSIGNMENT 9**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct data

{ int n;

struct data \*node;

}\*f=NULL;

int insertvalid(struct data \* a,int i)

{ struct data \*p=f,\*q=f; int j;

for( j=0;j<i;j++)

{ p=p->node;

q=q->node; }

q=q->node;

p->node=a;

a->node=q;

return 0;

}

void display()

{ struct data \*i; i=f;

while(i!=NULL)

{ printf(" %d - > ",i->n);

i=i->node; }}

int deletes()

{ int i,j; struct data \*q=f,\*r=f,\*p=NULL;

if(f==NULL)

{printf(" NO DATA FURTHER DELETION NOT POSSIBLE ");

return 0;}

scanf("%d",&i);

i=i-1;

if(i==0)

{ r=r->node;

free(f);

f=r;

display();

return 0;

}

for(j=1;j<i;j++)

{ q=q->node;

r=r->node;

}

r=r->node;

r=r->node;

p=q->node;

q->node=r;

free(p);

display();

return 0;

}

int reversed()

{ struct data \*a=f,\*b=NULL,\*c=f,\*d=NULL;

while(c!=NULL)

{ if(c->node==NULL)

break;

c=c->node; }

c->node=d;

b=c;

while(f!=c)

{ a=f;

f=f->node;

a->node=NULL;

b=b->node;

c->node=a;

a->node=b;

b=c; }

f=c;

return 0; }

int create()

{ int i;struct data \* p =(struct data \*)malloc(sizeof(struct data));

scanf("%d",&p->n);

p->node=NULL;

if(f!=NULL)

{ display();

printf(" \n\nENTER THE POSITION TO INSERT NEW DATA \ncount starts from 1");

scanf("%d",&i);

if(i==1)

{ p->node=f;

f=p;

return 0;

}

insertvalid(p,i-2);

}

if(f==NULL)

{ f=p;

printf("FIRST DATA CREATED AND INSERTED ");

}

return 0;

}

int main()

{ char a;

ab:

printf(" \n\n\n\t\t\t<< MENU >>\n\n1.CREATE AND INSERT\n2.DELETE \n3.DISPLAY\n4.REVERSE DISPLAY\n5.LIST REVERT\n\n\t\t");

a=getch();

switch(a)

{

case '1':

create();

goto ab;

case '2':

deletes();

goto ab;

case '3':

display();

goto ab;

case'4':

reversed();

display();

reversed();

goto ab;

case '5':

reversed();

printf("\n\n");

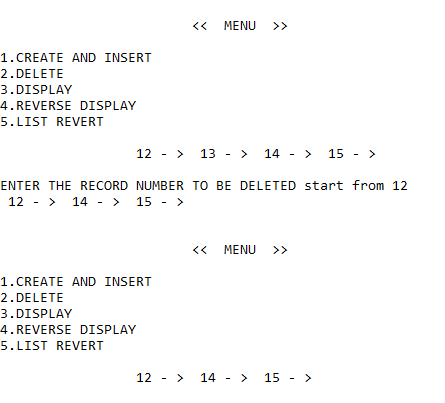
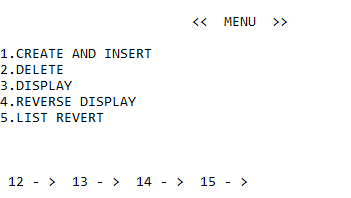
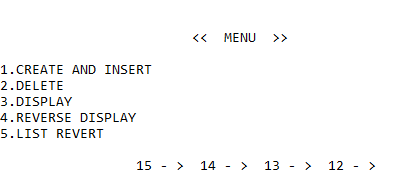
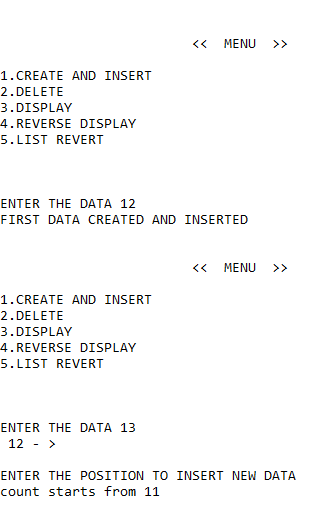
display();

goto ab;

}

return 0;

}



**ASSIGNMENT 10**

#include<conio.h>

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

struct cll

{

float coff;

int power;

struct cll\* next;

};

struct cll\* val (struct cll \*a ,struct cll \*node)

{ struct cll \*u=NULL;

if(a==NULL)

{

a=node;

a->next=a;

return a;

}

if(node->power > a->power)

{

node->next=a->next;

a->next=node;

a=node;

return a;

}

if(node->power == a->power)

{

a->coff=node->coff+a->coff;

return a;

}

u=a;

while(u->next!=a && node->power>=u->next->power)

u=u->next;

if(node->power == u->power)

u->coff = u->coff+node->coff;

else

{

node->next=u->next;

u->next=node;

}

return a;

}

struct cll\* insert(struct cll \*a,float co,float po)

{

struct cll \*node;

node=(struct cll\*)malloc(sizeof(struct cll));

node->coff=co;

node->power=po;

node->next=NULL;

a=val(a,node);

return a;

}

void display(struct cll \* q)

{

struct cll \*u;

u=q;

u=q->next;

do

{

printf("%5.1f x^ %d + ",u->coff,u->power);

u=u->next;

}while(u!=q->next);

}

void evaluate(float a, struct cll \* b)

{ long int r=0;

struct cll \*c=b;

do

{

r=r+(c->coff)\*pow(a,c->power);

c=c->next;

}while(b->power!=c->power);

printf("\n\n ANSWER IS : - %lld",r);

}

int sum(struct cll \*a, struct cll \* b)

{

struct cll \*c=NULL;

struct cll \*u=NULL,\*v=NULL;

u=a->next;

do

{v=(struct cll \*)malloc(sizeof(struct cll));

v->coff=u->coff;

v->power=u->power;

v->next=NULL;

c=val(c,v);

u=u->next;

}while(u!=a->next);

u=b->next;

do

{v=(struct cll \*)malloc(sizeof(struct cll));

v->coff=u->coff;

v->power=u->power;

v->next=NULL;

u=u->next;

c=val(c,v);

}while(u!=b->next);

display(c);

return 0;

}

int multiply(struct cll \*a, struct cll \*b)

{ struct cll \*h1=0,\*h2=0,\*h3=NULL,\*r=0;

h2=b->next;

do

{

h1=a->next;

do

{

r=(struct cll \*)malloc(sizeof(struct cll));

r->power=(h1->power+h2->power);

r->coff=(h1->coff\*h2->coff);

r->next=NULL;

h3=val(h3,r);

h1=h1->next;

}while(h1!=a->next);

h2=h2->next;

}while(h2!=b->next);

display(h3);

return 0;

}

int main()

{ struct cll \*p=NULL,\*q=NULL;

char a;

float co=0,de=0,qa;

int ra;

ab:

printf("\n\n\n\t\t\t..:: MENU ::.. \n\n");

printf("\n1.inset in 1st polynomial\n2.insert in second polynomial\n");

printf("\n3.print polynomial\n4.add \n5.multiply\n6.evaluate\n\n\t\t");

a=getch();

switch(a)

{

case'1':

{ printf("\n\nENTER THE COFFECIENT - ");

scanf("%50f",&co);

printf("\nENTER THE DEGREE - ");

scanf("%50f",&de);

p=insert(p,co,de);

goto ab;

}

case'2':

{printf("\n\nENTER THE COFFECIENT - ");

scanf("%50f",&co);

printf("\nENTER THE DEGREE - ");

scanf("%50f",&de);

q=insert(q,co,de);

goto ab;

}

case'3':

if(p!=NULL)

{printf(" EQUATION 1 : - \n\n");

display(p);}

if(q!=NULL)

{printf(" EQUATION 2 : - \n\n");

display(q);}

goto ab;

case'4':

sum(p,q);

goto ab;

case'5':

multiply(p,q);

goto ab;

case'6':

printf("\n\nENTER A VARIABLE & POLYNOMIAL NUMBER TO EVALUATE - ");

scanf("%f%d",&qa,&ra);

if(ra==1 && p!=NULL)

evaluate(qa,p);

else

if(q!=NULL)

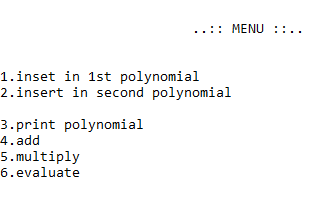
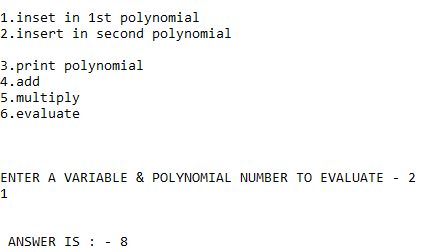
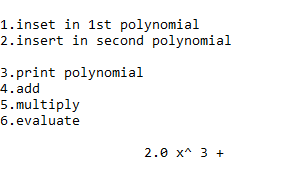
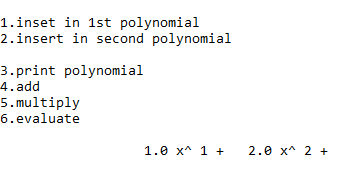
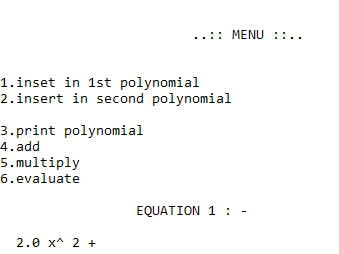
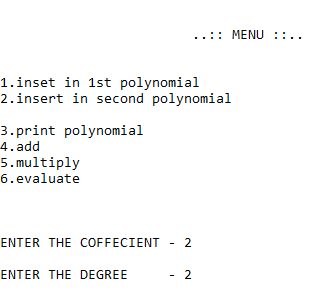
evaluate(qa,q);

goto ab;

}

return 0;

}



**ASSIGNMENT 11**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct data

{

int n;

struct data \*node;

struct data \*node2;

}

\*f=NULL,\*b=NULL;

void modify()

{ int q,i;

struct data \*a;

a=f;

printf("ENTER THE DATA NUMBER TO BE MODIFIED ");

scanf("%d",&q);

for(i=1;i<q;i++)

a=a->node;

scanf("%d",&a->n);

}

void displayb ()

{ struct data \*i;

i=b;

while(i!=NULL)

{

printf(" %d - > ",i->n);

i=i->node2;

}

}

void displayforward ()

{ struct data \*i;

i=f;

while(i!=NULL)

{

printf(" %d - > ",i->n);

i=i->node;

}

}

int deletes()

{ int i,j;

struct data \*q=f,\*r=f,\*p=NULL;

displayforward();

if(f==NULL)

{ printf(" NO DATA FURTHER DELETION NOT POSSIBLE ");

return 0;

}

printf("\n\nENTER THE RECORD NUMBER TO BE DELETED start from 1");

scanf("%d",&i);

if(i==1)

{

r=r->node;

free(f);

f=r;

displayforward();

return 0;

}

i=i-1;

for(j=1;j<i;j++)

{

q=q->node;

r=r->node;

printf("%d",r->n);

if(r->node->node==NULL)

{

p=b;

b=b->node2;

b->node=NULL;

p->node2=NULL;

free(p);

return 0;

}

}

r=r->node;

r=r->node;

p=q->node;

p->node=NULL;

p->node2=NULL;

q->node=r;

r->node2=q;

free(p);

displayforward();

return 0;

}

int intsert(struct data \*r)

{

b->node=r;

r->node2=b;

b=r;

return 0;

}

int insertvalid(struct data \*r,int t)

{ int i;

struct data \*a=f,\*c=f;

for( i=1;i<t;i++)

{ a=a->node;

c=c->node;

if(a==b)

{

intsert(r);

return 0;

}

}

c=c->node;

a->node=r;

r->node=c;

c->node2=r;

r->node2=a;

return 0;

}

int create()

{

struct data \* p;

int i;

p=(struct data \*)malloc(sizeof(struct data));

printf("\n\nENTER THE DATA ");

scanf("%d",&p->n);

p->node=NULL;

p->node2=NULL;

if(f!=NULL)

{

displayforward();

printf(" \n\nENTER THE POSITION TO INSERT NEW DATA \ncount starts from 1");

scanf("%d",&i);

if(i==1)

{

f->node2=p;

p->node=f;

f=p;

return 0;

}

insertvalid(p,i-1);

}

if(f==NULL)

{

f=p;

b=p;

printf("FIRST DATA CREATED AND INSERTED ");

}

return 0;

}

int main()

{ char a;

ab:

printf(" \n\n\n\t\t\t<< MENU >>\n\n1.CREATE AND INSERT\n2.DELETE \n3.DISPLAY FORWARD\n4.DISPLAY BACKWARD\n5.MODIFY\n\n\t\t");

a=getch();

switch(a)

{

case '1':

create();

goto ab;

case '2':

deletes();

goto ab;

case '3':

displayforward();

goto ab;

case'4':

displayb();

goto ab;

case '5':

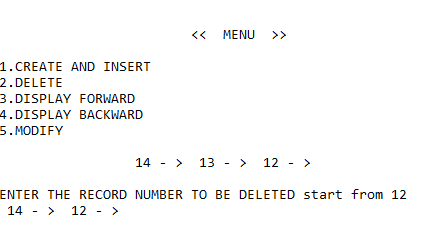
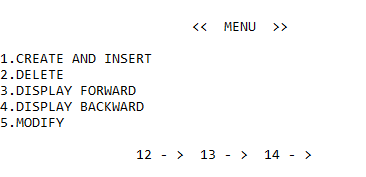
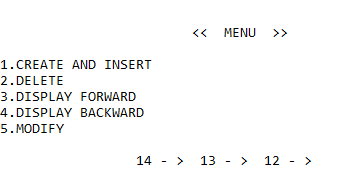
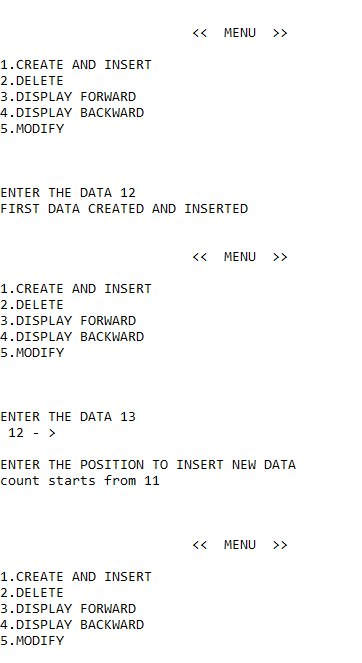
modify();

goto ab;

}

return 0;

}



**ASSIGNMENT 12**

#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<stdlib.h>

struct node

{ char data[100];

struct node\* left,\*right;

}\*start=NULL,\*cord=NULL;

int create()

{ struct node \*p=(struct node \*)malloc(sizeof(struct node));

printf("\n\nENTER THE DATA - ");

scanf("%s",&p->data);

fflush(stdin);

p->left=NULL;

p->right=NULL;

if(start==NULL)

{start=p;

return 0; }

printf("\n\nclick (l) for sub-data or (r) for main data - ");

char t;

t=getch();

fflush(stdin);

if(t=='r')

{

cord=start;

while(cord->right!=NULL)

cord=cord->right;

cord->right=p;}

if(t=='l')

{

cord=start;

char i[100];

scanf("%s",&i);

while(strcmp(cord->data,i))

{printf("%d",cord->data);

cord=cord->right;}

if (!strcmp(cord->data,i))

{

if(cord->left==NULL)

{cord->left=p;

return 0;

}

else

{

struct node \*o;

o=cord->left;

cord->left=p;

p->right=o; }

}

else

{

printf("NO DATA IS THERE - ");

free(p);

}

}

}

void display(struct node \*r)

{ while(cord!=NULL)

{printf("%s\n",cord->data);

cord=cord->right;}}

int sub(struct node\*r)

{

if(r==NULL)

return 0;

printf("%s\n",r->data);

sub(r->left);

printf("\n\n");

sub(r->right);

}

int main()

{ab:

printf("\n\n\n\t\t\t..:: MENU ::..\n\n");

printf("1.insert\n\n2.display main list \n\n3.display sublist if any \n\n");

char i;

i=getch();

switch(i)

{

case'1':

create();

goto ab;

case'2':

display(start);

goto ab;

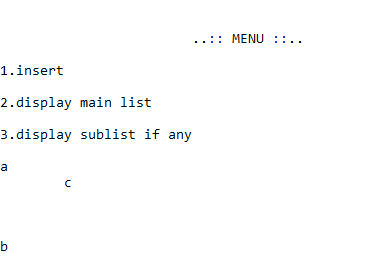
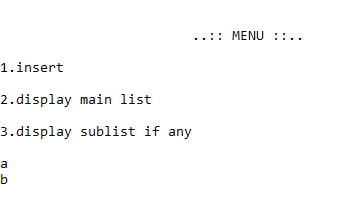
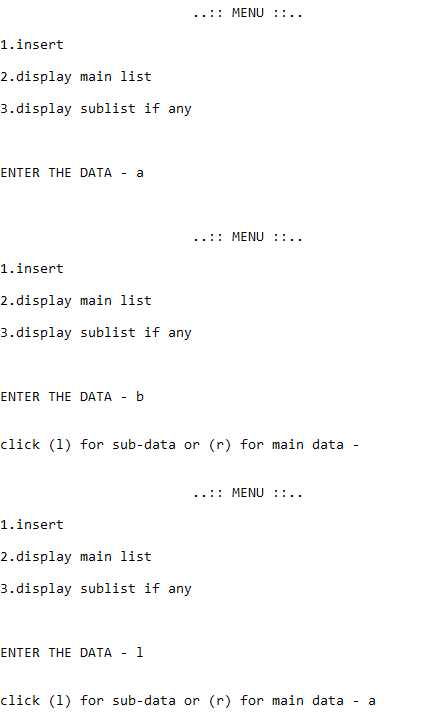
case '3':

sub(start);

goto ab;

}

return 0;

}

void fasttrans(matrix \*\*matA, matrix \*\*matB, int col)

{

int rowterm[10] = {0};

int startingpos[col];

int i,j;

matB[0].row = matA[0].col;

matB[0].col = matA[0].row;

matB[0].value = matA[0].value;

if(matA[0].value > 0){

for(i = 1; i < matA[0].value; i++)

rowterm[matA[i].col]++;

startingpos[0]=1;

for(i = 1; i < matA[0].col; i++)

startingpos[i] = startingpos[i-1] + rowterm[i-1];

for(i = 1; i <= matA[0].value; i++){

j = startingpos[matA[i].col]++;

matB[j].row = matA[j].col;

matB[j].col = matA[j].row;

matB[j].value = matA[j].value;

}

}

}

