**IFTM UNIVERSITY**

**SCHOOL OF COMPUTER SCIENCE AND APPLICATIONS**

# DEPARTMENT OF COMPUTER APPLICATIONS



# LAB MANNUAL

On

**Data Structure Lab (“P020717P”)**

**Session:2024-25**

# 

**Submitted TO Submitted By**

**Dr. Lalit Johari (Student name)**

**LIST OF PROGRAMS**

**Unit 1 : Assignment**

1. To find both the largest and smallest number in a list of integers
2. To determine if the given string is a palindrome or not.
3. Write a program for finding the roots of a quadratic equation using function.
4. Write a program for printing a string input by a user in reverse order.
5. Write a program to add two matrices.
6. Write a program for finding the value of n! by using recursive function. The value of n to be given by the user.
7. Write a program for finding the GCD of two numbers using recursive function.
8. Write a program for printing the nth Fibonacci number by using recursive function.

**Unit 2 : Assignment**

1. Write a program for implementing Stack operations **PUSH** and **POP** using array.
2. Write a program for implementing Stack operations **PUSH** and **POP** using linked list.
3. Write a program for implementing Queue operations using Array.
4. Write a program for showing Queue operations using linked list.
5. Write a program to convert infix expression to postfix expression using stack.
6. Write a program for evaluating postfix expression.
7. Write a program for printing the moves in Tower of Hanoi problem when there are **n** disks.

**Unit 3: Assignment**

1. Write a program to implement a singly linked list.
2. Write a program to implement doubly linked list.
3. Write a program for concatenate two linked lists.

**Unit 4 : Assignment**

1. Write a program for printing pre-order traversal of Binary Tree.
2. Write a program for printing post-order traversal of Binary Tree.
3. Write a program for printing in-order traversal of Binary Tree.
4. Write a program for implementing Binary Search Tree and various operations on it.

**Unit 5 : Assignment**

1. Write a program for Linear Search.
2. Write a program for Binary Search.
3. Write a program for implementing Bubble Sort.
4. Write a program for implementing Heap sort.
5. Write a program for implementing Quick sort.
6. Write a program for implementing Merge sort.
7. Write a program for implementing Insertion sort.
8. Write a program for implementing Radix Sort.
   1. **To find both the largest and smallest number in a list of integers**

Step 1: Start

Step 2: read n  
Step 3: initialize i=0  
Step 4: if i<n do as follows. If not goto step 5

Read a[i]

Increment i

Goto step 4  
Step 5: min=a[0], max=a[0]  
Step 6: initialize i=0  
Step 7: if i<n do as follows. If not goto step 8

If a[i]<min  
Assign min=a[i]  
Increment i goto Step 7

Step 8: print min,max

Step 9: stop

**Program:**

#include<stdio.h>

void main()

{  
int a[10],i,n,min,max;  
clrscr();  
printf("enter the array size:");  
scanf("%d",&n);  
printf("Enter the elements of array");  
for(i=0;i<n;i++) // read the elements of an array  
scanf("%d",&a[i]);  
min=a[0];  
max=a[0];  
for(i=0;i<n;i++)// read the elements of an array  
{

if(a[i]<min)// check the condition for minimum value  
min=a[i];  
if(a[i]>max)//check the condition for maximum value  
max=a[i];

}  
printf("maximum value is:%d\n",max);  
printf("minimum value is:%d\n",min);  
getch();

}Output:

1.enter the array size:4

Enter the elements of array 36 13

2

45

maximum value is:45

minimum value is:2

2.enter the array size:5

Enter the elements of array 6 2 1 8  
maximum value is:8  
minimum value is:1

3.enter the array size:5

Enter the elements of array-6 9 -9

2

5

maximum value is:9

minimum value is:-9

**2)To determine if the given string is a palindrome or not**

Step 1:start  
Step 2: read the string  
Step 3: store reverse of the given string in a temporary string  
Step 4: compare the two strings  
Step 5: if both are equal then print palindrome  
Step 6: otherwise print not palindrome  
Step 7: stop

**Program:**

#include<stdio.h>

#include<string.h>

enum Boolean{false,true};  
enum Boolean IsPalindrome(char string[])  
{

int left,right,len=strlen(string);

enum Boolean matched=true;

if(len==0)  
return 0;  
left=0;

right=len-1;

/\* Compare the first and last letter,second & second last & so on \*/  
while(left<right&&matched)  
{

if(string[left]!=string[right])  
matched=false;  
else

{

left++;

right--;

}

}

return matched;

}

int main()

{  
char string[40];  
clrscr();  
printf("\*\*\*\*Program to test if the given string is a palindrome\*\*\*\*\n");  
printf("Enter a string:");  
scanf("%s",string);  
if(IsPalindrome(string))  
printf("The given string %s is a palindrome\n",string);  
else  
printf("The given string %s is not a palindrome\n",string);  
getch();  
return 0;  
}

**Output:**

1. Enter the string:malayalam

The given string malayalam is a palindrome

2. Enter the string:india

The given string india is not a palindrome

**3)Write a program for finding the roots of a quadratic equation using function.**

Step 1: start  
Step 2: read the a,b,c value  
Step 3: if b\*b-4ac>0 then

Root 1= (-b+ pow((b\*b-4\*a\*c),0.5))/2\*a

Root 2= (-b-pow((b\*b-4\*a\*c),0.5))/2\*a

Step 4: if b\*b-4ac=0 then

Root1 = Root2 = -b/(2\*a)  
Step 5: Otherwise Print Imaginary roots. Goto step 7.  
Step 6: print roots  
Step 7: stop

Program:

#include<stdio.h>  
#include<math.h>  
void main()  
{float a,b,c,r1,r2,d;

clrscr();  
printf("Enter the values for equation:");  
scanf("%f%f%f",&a,&b,&c);  
/\* check the condition \*/  
if(a==0)  
printf("Enter value should not be zero ");  
else  
{

d=b\*b-4\*a\*c;

/\* check the condition \*/

if(d>0)

{

r1=(-b+sqrt(d)/(2\*a));  
r2=(-b-sqrt(d)/(2\*a));  
printf("roots are real and unequal\n");  
printf("%f\n%f\n",r1,r2);

}else

if(d==0)

{

r1=-b/(2\*a);  
r2=-b/(2\*a);  
printf("roots are real and equal\n");  
printf("root=%f\n",r1);  
printf("root=%f\n",r2);

}

else

printf("roots are imaginary");

}getch();

}Output:

1. Enter the values for equation: 1, 6, 9  
Roots are real and equal  
Root= -3.0000  
Root= -3.0000

2. Enter the values for equation: 2, 7, 6  
Roots are real and unequal  
Root= -6.75  
Root= -7.25

3. Enter the values for equation: 1, 2, 3

Roots are imaginary

**4)Write a program for printing a string input by a user in reverse order.**

#include <stdio.h>  
   
void reverse (char \*s)  
{

char \*t = s;  
   
 while (\*t != '\0') t++;  
 while (s < t)  
 {  
 int c = \*s;  
 \*s++ = \*--t;  
 \*t = c;  
 }  
}  
   
int main ()  
{  
 char text1[] = "asdf", text2[] = "";  
 reverse (text1);  
 [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html) ("'%s'\n", text1);  
 reverse (text2);  
 [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html) ("'%s'\n", text2);  
   
 return 0;  
}

**5)Write a program to addition of two matrices.**

algorithm:

Step 1: start  
Step 2: read the size of matrices A,B – m,n  
Step 3: read the elements of matrix A  
Step 4: read the elements of matrix B  
Step 5: select the choice for you want. If you select case 1 then goto matric

addition. Else goto Step 7.  
Step 6: print Sum of matrix A and B  
Step 7: if you select case 2 then goto matrix multiplication  
Step 8: check if n=p, if not print matrices can not be multiplied  
Step 9: Otherwise perform the multiplication of matrices  
Step 10: Print the resultant matrix  
Step 11: Stop

Program:

#include<stdio.h>

void main()

{  
int ch,i,j,m,n,p,q,k,r1,c1,a[10][10],b[10][10],c[10][10];  
clrscr();  
printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
printf("\n\t\tMENU");  
printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
printf("\n[1]ADDITION OF TWO MATRICES");  
printf("\n[2]MULTIPLICATION OF TWO MATRICES");  
printf("\n[0]EXIT");  
printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
printf("\n\tEnter your choice:\n");  
scanf("%d",&ch);

if(ch<=2 & ch>0)

{

printf("Valid Choice\n");

}switch(ch)

{

case 1:  
printf("Input rows and columns of A & B Matrix:");  
scanf("%d%d",&r1,&c1);  
printf("Enter elements of matrix A:\n");  
for(i=0;i<r1;i++)  
{

for(j=0;j<c1;j++)

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

}printf("Enter elements of matrix B:\n");

for(i=0;i<r1;i++)

{

for(j=0;j<c1;j++)

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

}  
printf("\n= == = =Matrix Addition=====\n");  
for(i=0;i<r1;i++)

{

For(j=0;j<c1;j++)

printf("%5d",a[i][j]+b[i][j]);

52

printf("\n");

}

break;

case 2:  
printf("Input rows and columns of A matrix:");  
scanf("%d%d",&m,&n);  
printf("Input rows and columns of B matrix:");  
scanf("%d%d",&p,&q);  
if(n==p)

{  
printf("matrices can be multiplied\n");  
printf("resultant matrix is %d\*%d\n",m,q);  
printf("Input A matrix\n");  
read\_matrix(a,m,n);  
printf("Input B matrix\n");  
/\*Function call to read the matrix\*/  
read\_matrix(b,p,q);  
/\*Function for Multiplication of two matrices\*/  
printf("\n= == = =Matrix Multiplication=====\n");  
for(i=0;i<m;++i)

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

{

c[i][j]=0;

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

c[i][j]=c[i][j]+a[i][k]\*b[k][j];

}

printf("Resultant of two matrices:\n");

write\_matrix(c,m,q);

}/\*end if\*/

else

{

printf("Matrices cannot be multiplied.");  
}  
/\*end else\*/  
break;

case 0:  
printf("\n Choice Terminated");  
exit();  
break;

default:

printf("\n Invalid Choice");

}getch();

}/\*Function read matrix\*/

int read\_matrix(int a[10][10],int m,int n)

{  
int i,j;  
for(i=0;i<m;i++)

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

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

return 0;

}

/\*Function to write the matrix\*/  
int write\_matrix(int a[10][10],int m,int n)  
{

int i,j;

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

{  
for(j=0;j<n;j++)  
printf("%5d",a[i][j]);  
printf("\n");

}

return 0;

}

Output:

1.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MENU

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
[1]ADDITION OF TWO MATRICES  
[2]MULTIPLICATION OF TWO MATRICES  
[0]EXIT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter your choice:

1Valid Choice

Input rows and columns of A & B Matrix:2

2Enter elements of matrix A:

222

54

2Enter elements of matrix B:

2222=====Matrix Addition=====

4 4

4 4

**6) Write a program for finding the value of n! by using recursive function. The value of n to be given by the user.**

Step 1: start

Step 2: read n  
Step 3: call sub program as f=fact(n)  
Step 4: print f value  
Step 5: stop

Sub program:

Step 1: initialize the f  
Step 2: if n= = 0 or n == 1 return 1 to main program if not goto step 3  
Step 3: return n\*fact(n-1) to main program

Program:

#include<stdio.h>  
#include<conio.h>  
int fact(int n)

{  
int f;  
if((n==0)||(n==1)) // check the condition for the n value  
return(n);  
else  
f=n\*fact(n-1); //calculate the factorial of n  
return(f);

}

void main()

{  
int n;  
clrscr();  
printf("enter the number :");  
scanf("%d",&n);

printf("factoria of number%d",fact(n));

getch();

}

Output:

1. Enter the number : 5

Factorial of number: 120

2. Enter the number : 3

Factorial of number: 6

3. Enter the number : 9

Factorial of number: -30336

1. **Write a program for finding the GCD of two numbers using recursive function.**

Algorithm: main program

Step 1: start  
Step 2: read a,b  
Step 3: call the sub program GCD(a,b) for print the value  
Step 4: stop

Sub program:

Step 1: if n>m return GCD(n,m)  
Step 2: if n==0 return m else goto step 3  
Step 3: return GCD (n,m%n)  
Step 4: return to main program

Program:

#include<stdio.h>  
#include<conio.h>  
int gcdrecursive(int m,int n) // starting of the sub program  
{

if(n>m)

return gcdrecursive(n,m);  
if(n==0)  
return m;

else

return gcdrecursive(n,m%n); // return to the main program

}void main()

{

int a,b,igcd;

clrscr();  
printf("enter the two numbers whose gcd is to be found:");  
scanf("%d%d",&a,&b);  
printf("GCD of a,b is %d",gcdrecursive(a,b)); // return to the sub program  
getch();

}

Output:

1. enter the two numbers whose gcd is to be found:5,25

GCD of a,b is : 5

2. enter the two numbers whose gcd is to be found:36,54

GCD of a,b is : 18

3. enter the two numbers whose gcd is to be found:11,13

GCD of a,b is : 1

1. **Write a program to print the Fibonacci series for 1 to n value.**

Step 1: start  
Step 2: initialize the a=0, b=1  
Step 3: read n  
Step 4: if n== 1 print a go to step 7. else goto step 5  
Step 5: if n== 2 print a, b go to step 7 else print a,b  
Step 6: initialize i=3

if i<= n do as follows. If not goto step 7  
c=a+b  
print c  
a=b  
b=c  
increment I value  
goto step 6(i)

Step 7: stop

**Program:**

#include<stdio.h>

void main()

{  
int a,b,c,n,i;  
clrscr();  
printf("enter n value");  
scanf("%d",&n);  
a=0;  
b=1;  
if(n==1)  
printf("%d",a);  
else  
if(n==2)  
printf("%d%d",a,b);  
else

{

printf("%d%d",a,b);  
//LOOP WILL RUN FOR 2 TIME LESS IN SERIES AS THESE WAS  
PRINTED IN ADVANCE

for(i=2;i<n;i++)

{  
c=a+b;  
printf("%d",c);  
a=b;  
b=c;

}

getch();

}

}Output:

1. Enter n value : 5

01 1 2 3

2. Enter n value : 7

0 112358

3. Enter n value : -6

0 1

1. **Write a program for printing the moves in Tower of Hanoi problem when there are n disks.**

Step 1: start  
Step 2: initialize the source=a, intermediate=c, destination = d  
Step 3: read n  
Step 4: call the sub program Hanoi recursion (n value,a ,b, c)  
Step 5: stop  
Sub program:

Step 1: if n== 1 call the sub program Hanoi recursion (num-1, a, c, b)  
Step 2: print the output from a to b  
Step 3: call the sub program Hanoi recursion(num-1, b, c, a)  
Step 4: return to main program

Program:

#include<stdio.h>  
#include<conio.h>  
void Hanoirecursion(int num,char ndl1,char ndl2,char ndl3)  
{

if(num==1)

{  
printf("Move top disk from needle %c to needle %c",ndl1,ndl2);  
return;

}  
Hanoirecursion(num-1,ndl1,ndl3,ndl2);  
printf("Move top dis from needle %c to needlle %c",ndl1,ndl2);  
Hanoirecursion(num-1,ndl3,ndl2,ndl1);

}void main()

{  
int no;  
clrscr();  
printf("Enter the no. of disk to be transferred:");  
scanf("%d",&no);  
if(no<1)  
printf("\n There's nothing to move");  
else  
printf("\n recursive");  
Hanoirecursion(no,'A','B','C');  
getch();

}Outputs:

1. Enter the no. of disk to be transferred :3  
Move top disk from needle a to needle b  
Move top disk from needle a to needle c  
Move top disk from needle b to needle c  
Move top disk from needle a to needle b  
Move top disk from needle c to needle a  
Move top disk from needle c to needle b  
Move top disk from needle a to needle b

1. **Write a program for implementing Stack operations PUSH and POP using linked list.**

**Algorithm:**

Step 1: Start  
Step 2: Declare the structure for the stack pointers.  
Step 3: Define the push function  
Step 4: Define the pop function  
Step 5: Define the display function  
Step 6: Read the choice  
Step 7: if choice = push

Create a cell for the TOP cell in the stack.  
Place the date in the TOP cell  
Place the TOP pointer to the new cell

S Step 8: if choice=pop  
Check if empty stack. If so, print stack is empty.  
Otherwise, remove the TOP cell.

Step 9: if choice=display

Display all the elements in the Stack.

Step 10: Stop

Program:

#include<stdio.h>

#include<conio.h>

struct st\_point

{  
int ele;  
struct st\_point \*l;

}\*t;

int i;

void push\_ele(int j);  
int pop\_ele();  
void display\_ele();

void main()

{  
char choice,num1=0,num2=0;  
int i;  
while(1)

{  
clrscr();  
printf("======================================");  
printf("\n\t\t MENU ");  
printf("\n======================================");  
printf("\n[1] Using Push Function");  
printf("\n[2] Using Pop Function");  
printf("\n[3] Elements present in Stack");  
printf("\n[4] Exit\n");  
printf("\n\tEnter your choice: ");  
fflush(stdin);  
scanf("%c",&choice);

switch(choice-'0')

{

case 1:

{  
printf("\n\tElement to be pushed:");  
scanf("%d",&num1);  
push\_ele(num1);  
break;

}case 2:

{  
num2=pop\_ele(1);  
printf("\n\tElement to be popped: %d\n\t",num2);  
getch();  
break;

}case 3:

{  
printf("\n\tElements present in the stack are:\n\t");  
display\_ele();  
getch();

break;

}case 4:

exit(1);

break;

default:  
printf("\nYour choice is invalid.\n");  
break;

}

}

}/\*Inserting the elements using push function\*/

void push\_ele(int j)

{  
struct st\_point \*m;  
m=(struct st\_point\*)malloc(sizeof(struct st\_point));  
m->ele=j;  
m->l=t;  
t=m;  
return;

}/\*Removing the elements using pop function\*/

int pop\_ele()

{

if(t==NULL)

{  
printf("\n\STACK is Empty.");  
getch();  
exit(1);

}else

{  
int i=t->ele;  
t=t->l;  
return (i);

}  
return 0;  
}/\*Displaying the elements \*/

void display\_ele()

{

struct st\_point \*pointer=NULL;

pointer=t;  
while(pointer!=NULL)  
{

printf("%d\t",pointer->ele);

pointer=pointer->l;

}

}Output:

======================================

MENU

======================================  
[1] Using Push Function  
[2] Using Pop Function  
[3] Elements present in Stack  
[4] Exit

Enter your choice: 1

Element to be pushed:23

1. **Write a program for implementing Queue operations using Array.**

**ALGORITHM FOR INSERTING AN ELEMENT IN TO A QUEUE:**

Function QINSERET(Q,F,R,N,Y)

Step 1: [overflow]  
If R>=N  
Then printf(“ overflow”)

Return

Step 2: [Increment rear pointer]

R<-R+1

Step 3: [ Insert element]

Q[R]<-y

Step 4: [Is front pointer properly set?]  
If F=0  
Then f<-1  
Return

**ALGORITHM FOR DELETING AN ELEMENT FROM A QUEUE:**

Function QDELETE(Q,F,R)

Step 1: [Underflow]  
If F=0  
Then printf(“Queue underflow”)  
Return(0)

Step 2: [Delete element]

y<-q[f]

Step 3: [Is Queue Empty?]

If F=R

Then F=R=0

Else

F=F+1

Step 4:[Return element]

Return(r)

Program:

# include <stdio.h>  
# define size 4  
int front=0,rear=-1,item,choice,a[size];  
main()  
{clrscr();

while(1)

{

printf("\*\*\* MENU \*\*\*\n 1. INSERTION\n 2. DELETION\n

3.TRAVERSE\n 4. EXIT\n");  
printf("enter your choice:");  
scanf("%d",&choice);  
switch(choice)  
{

case 1:insertion();

break;

case 2:deletion();

break;

case 3:traverse();

break;  
case 4:exit();  
default:printf("\*\*\* wrong choice \*\*\*\n");

}

}getch();

}insertion()

{  
if(rear==size-1)  
printf("\*\*\* queue is full \*\*\*\n");  
else  
{

printf("enter item into queue:");  
scanf("%d",&item);  
rear++;  
a[rear]=item;

}

}deletion()

{  
if(front==rear+1)  
printf("\*\*\* queue is empty \*\*\*\n");  
else  
{

item=a[front];  
front++;  
printf("the deleted item from queue is %d\n",item);  
}

}traverse()

{  
int i;  
if(front==rear+1)  
printf("\*\*\* queue is empty \*\*\*\n");  
else

{  
for(i=front;i<=rear;i++)  
if(i==front && rear==i)  
printf("%d at %d ->front=rear\n",a[i],i);  
else  
if(i==rear)  
printf("%d at %d ->rear\n",a[i],i);  
else  
if(i==front)  
printf("%d at %d ->front\n",a[i],i);  
else  
printf("%d at %d\n",a[i],i);

}

}

Input/Output:

\*\*\* MENU \*\*\*  
1. INSERTION  
2. DELETION  
3. TRAVERSE  
4. EXIT

enter your choice:1

enter item into queue:11

\*\*\* MENU \*\*\*  
1. INSERTION  
2. DELETION  
3. TRAVERSE  
4. EXIT

1. **Write a program for Linear Search.**

1. Start  
2. Read the value of n  
3. for i=1 to n increment in steps of 1

Read the value of ith element into array  
4. Read the element(x) to be searched  
5. search<--linear(a,n,x)  
6. if search equal to 0 goto step 7 otherwise goto step 8  
7. print unsuccessful search  
8. print successful search  
9. stop

LINEAR FUNCTION

1. start  
2. for i=1 to n increment in steps of 1  
3. if m equal to k[i] goto step 4 otherwise goto step 2  
4. return i  
5. return 0  
6. stop

Program:

#include<stdio.h>  
main()  
{int i,j,n,a[10],key;

clrscr();  
printf("enter range for array:");  
scanf("%d",&n);  
printf("enter elements into array:");  
for(i=0;i<=n;i++)  
scanf("%d",&a[i]);  
printf("enter the search element:");  
scanf("%d",&key);  
for(i=0;i<=n;i++)  
{if(key==a[i])

{printf("element %d found at %d",key,i);

break;

}else

if(i==n)  
printf("element %d not found in array",key);  
}getch();

}

Input/Output:

enter range for array:4  
enter elements into array:56  
43  
12  
88  
9

enter the search element:9

element 9 found at 4

1. **Write a program for Binary Search.**

1. Start  
2. Read the value of n  
3. for i=1 to n increment in steps of 1

Read the value of ith element into array  
4. Read the element(x) to be searched  
5. search<--binary(a,n,x)  
6. if search equal to 0 goto step 7 otherwise goto step 8  
7.print unsuccessful search  
8. print successful search  
9. stop

**BINARY SEARCH FUNCTION**

1. start  
2. initialise low to 1 ,high to n, test to 0  
3. if low<= high repeat through steps 4 to 9 otherwise goto step 10

4. assign (low+high)/2 to mid

5. if m<k[mid] goto step 6 otherwise goto step 7

6. assign mid-1 to high goto step 3

7. if m>k[mid] goto step 8 otherwise goto step 9

8. assign mid+1 to low

9. return mid  
10. return 0  
11.stop

Program:

#include<stdio.h>  
main()  
{int i,j,n,a[10],key;

clrscr();  
printf("enter range for array:");  
scanf("%d",&n);  
printf("enter elements into array:");  
for(i=0;i<=n;i++)  
scanf("%d",&a[i]);  
printf("enter the search element:");  
scanf("%d",&key);  
for(i=0;i<=n;i++)  
{if(key==a[i])

{printf("element %d found at %d",key,i);

break;

}else

if(i==n)  
printf("element %d not found in array",key);  
}getch();

}

Input/Output:

enter range for array:4  
enter elements into array:56  
43  
12  
88  
9enter the search element:9

element 9 found at 4

1. **Write a program to implement a singly linked list.(Create , insert, delete, & traverse)**

Step 1: Start  
Step 2: Declare a structure named linked-list  
Step 3: Declare the pointers next, first, fresh, ptr  
Step 4: Print main menu  
Step 5: Read choice  
Step 6: Switch(choice)  
Step 7: If(choice==1)

7.1 Assign fresh=malloc(size of (node))  
7.2 Read the element fresh->data  
7.3 Read the choice where to insert

7.4:Switch(choice)  
7.4.1: If choice==1  
7..4.2: Call the function IBegin()  
7.4.3: If choice==2  
7.4.4: Call the function Iend()  
7.4.5: If choice==3  
7.4.6: Call the function Imiddle()

Step 8: If(choice==2)

8.1: Read the position to delete

8.2: Switch(choice)  
8.2.1: If choice==1  
8..2.2: Call the function DBegin()  
8.2.3: If choice==2  
8.2.4: Call the function Dend()  
8.2.5: If choice==3  
8.2.6: Call the function Dmiddle()

Step 9: If choice==3

9.1 Call function view

114

Step 10: If choice==4

10.1 Exit()  
Step 11: Start insert function  
Step 12: If(first==null)  
Step 13: First->data=e  
Step 14: First->next=null  
Step 15: Else declare new node  
Step 16:fresh->data=e  
Step 17: If choice=1  
Step 18: frsh->next=first  
Step 19: first=fresh  
Step 20:if choice=2  
Step 21: ptr=first  
Step 22: ptr->next=fresh  
Step 23: fresh->next=full  
Step 24: If choice =3  
Step 25: Enter the position  
Step 26:at p-1 node  
Step 27: fresh->next= ptr->next  
Step 28: ptr->next=fresh  
Step 29: for delete function  
Step 30: If first!=null  
Step 31: Enter the position to delete  
Step 32: If choice=1

Step 33: d=first->data  
Step 34: first=first->next  
Step 35: if choice=2  
Step 36: ptr=first  
Step 37: Traverse to last node  
Step 38: d=ptr->next->data  
Step 39: ptr ->next=ptr->next->next  
Step 40: Print d value  
Step 41: for function view  
Step 42: for ptr=first and ptr!=null and ptr=ptr->next  
Step 43: Print ptr->data  
Step 44: End

Program:

# include<stdio.h>

# include<malloc.h>

int ch,i,n,j,p,item;

/\* VARIABLE DECLARATION \*/  
/\* START OF STRUCTURE DEFINITION \*/  
struct link

{  
int data;  
struct link \*next;

}\*start,\*new,\*l,\*l1,\*start1,\*t;  
/\* END OF STRUCTURE DEFINITION \*/  
/\* START OF MAIN FUNCTION \*/  
main()

{clrscr();

start=NULL;  
start1=NULL;  
printf("\*\*\*\* MENU\*\*\*\* ");  
printf("\n 1.Insertion\n 2.Deletion\n 3.Traverse\n 4.Search\n 5.Sort\n 6.Merge\n  
7.Reverse\n");  
while(1)  
{

printf("enter the choice:");  
scanf("%d",&ch);  
switch(ch)  
{case 1: insert();

break;

case 2: delete();

break;

case 3: traverse();

break;

case 4: search();

break;

case 5: sort();

break;

case 6: merge();

break;

case 7: reverse();

break;  
case 8:exit();  
}

}getch();

}/\* END OF MAIN FUNCTION \*/

/\* START OF INSERT FUNCTION \*/

insert()

{l=start;

printf("enter the item to be inserted:");  
scanf("%d",&item);  
new=malloc(sizeof(struct link));  
new->data=item;  
if(start==NULL)  
{

new->next=NULL;

start=new;

}else

{  
printf("1.start\n2.middle\n3.end\n");  
printf("enter the place to place the item:");

scanf("%d",&ch);  
if(ch==1)  
{

new->next=start;

start=new;

}if(ch==2)

{  
printf("enter the position to place item:");  
scanf("%d",&p);  
for(i=1;i<p-1;i++)  
l=l->next;  
new->next=l->next;  
l->next=new;

}if(ch==3)

{  
while(l->next!=NULL)  
l=l->next;

new->next=NULL;

l->next=new;

}}}/\* END OF INSERT FUNCTION \*/

/\* START OF DISPLAY FUNCTION \*/

traverse()

{  
if(start==NULL)  
printf("LIST IS EMPTY\n");  
else

{  
for(l=start;l->next!=NULL;l=l->next)  
if(l==start)  
printf("\nstart:%d->",l->data);  
else  
printf("\n%7d->",l->data);  
if(l->next==NULL)  
printf("\n last:%d->\n",l->data);

}

}/\* END OF DISPLAY FUNCTION \*/

/\* START OF DELETE FUNCTION \*/

delete()

{  
l=start;  
if(start==NULL)  
printf("NO ITEMS IN THE LIST\n");  
else  
{

printf("1.start\n2.middle\n3.end\n");  
printf("enter the place to delete the item:");  
scanf("%d",&ch);  
if(ch==1)  
{

item=start->data;  
printf("deleted item is:%d\n",item);  
start=start->next;

}if(ch==2)

{  
printf("enter the position to delete item:");  
scanf("%d",&p);  
if(l->next==NULL)  
{

item=l->data;  
printf("deleted item is:%d\n",item);  
l=start=NULL;

}

else

{  
for(i=1;i<p-1;i++)  
l=l->next;  
item=l->next->data;  
printf("deleted item is:%d\n",item);  
l->next=l->next->next;

}

}if(ch==3)

{  
if(l->next==NULL)  
{item=l->data;

printf("deleted item is:%d\n",item);  
l=start=NULL;  
}

else

{  
while(l->next->next!=NULL)  
l=l->next;  
item=l->next->data;  
printf("deleted item is:%d\n",item);  
l->next=NULL;  
l=l->next;

}}}}/\* END OF DELETE FUNCTION \*/

/\* START OF SEARCH FUNCTION \*/

search()

{int f=0;

printf("enter the search item:");  
scanf("%d",&item);  
if(start==NULL)  
printf("LIST IS EMPTY");  
else  
{

for(l=start,i=1;l!=NULL;l=l->next,i++)  
if(l->data==item)  
{

f=1;

break;

}if(f==1)

printf("item %d found at position :%d\n",item,i);  
else  
printf("item %d not found\n",item);

}}/\* END OF SEARCH FUNCTION \*/

/\* START OF SORT FUNCTION \*/

sort()

{int t;

if(start==NULL)  
printf("LIST IS EMPTY");  
else  
{

for(l1=start;l1->next!=NULL;l1=l1->next)

{  
for(l=start;l->next!=NULL;l=l->next)  
if(l->data > l->next->data)  
{t=l->data;

l->data=l->next->data;  
l->next->data=t;  
}

}printf("THE SORTED ORDER IS:");

for(l=start;l!=NULL;l=l->next)  
printf("%3d",l->data);  
}printf("\n");

}

/\* END OF SORT FUNCTION \*/  
/\* START OF MERGE FUNCTION \*/  
merge()

{printf("enter no of elements to be inserted in second list :");

scanf("%d",&n);

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

{  
l1=start1;  
printf("enter the item to be inserted:");  
scanf("%d",&item);  
new=malloc(sizeof(struct link));  
new->data=item;  
new->next=NULL;  
if(start1==NULL)  
start1=new;  
else  
{

printf("1.start\n2.middle\n3.end\n");  
printf("enter the place to place the item:");  
scanf("%d",&ch);  
if(ch==1)

{  
new->next=start1;  
start1=new;

}

if(ch==2)

{  
printf("enter the position to place item:");  
scanf("%d",&p);  
for(i=1;i<p-1;i++)  
l1=l1->next;  
new->next=l1->next;  
l1->next=new;

}

if(ch==3)

{  
while(l1->next!=NULL)  
l1=l1->next;  
l1->next=new;

}}}

if(start==NULL)  
start=start1;  
else  
{

l=start;  
while(l->next!=NULL)  
l=l->next;

for(l1=start1;l1->next!=NULL;l1=l1->next)

{  
l->next=l1;  
l=l->next;

}}printf(" \*\*\* LIST IS MERGED \*\*\* \n");

}  
/\* END OF MERGE FUNCTION \*/  
/\* START OF REVERSE FUNCTION \*/  
reverse()

{if(start==NULL)

printf("LIST IS EMPTY\n");

else

{  
l=start;  
l1=t=NULL;  
while(l!=NULL)  
{

l1=t;  
t=l;  
l=l->next;  
t->next=l1;

}start=t;

printf(" \*\*\* LIST IS REVERSED \*\*\*\n");

}

}/\* END OF REVERSE FUNCTION \*/

\*\*\*\*\*OUTPUT \*\*\*\*\*

\*\*\*\*M EN U \*\*\*\*

1.Insertion

2.Deletion  
3.Traverse  
4.Search  
5.Sort  
6.Merge

7.Reverse  
enter the choice:1  
enter the item to be inserted:1  
enter the choice:1  
enter the item to be inserted:2  
1.start  
2.middle  
3.end  
enter the place to place the item:1  
enter the choice:1  
enter the item to be inserted:3  
1.start  
2.middle  
3.end  
enter the place to place the item:3  
enter the choice:1  
enter the item to be inserted:4  
1.start  
2.middle  
3.end  
enter the place to place the item:2  
enter the position to place item:3  
enter the choice:3

start:2->  
1->  
4->

last:3->  
enter the choice:4  
enter the search item:4  
item 4 found at position :3  
enter the choice:6  
enter no of elements to be inserted in second list :3  
enter the item to be inserted:5  
enter the item to be inserted:6  
1.start  
2.middle  
3.end  
enter the place to place the item:1  
enter the item to be inserted:7

1. **Write a program for implementing Stack operations PUSH and POP using** **array.**

**ALGORITHM FOR INSERTING AN ELEMENT IN A STACK:**

Function Push(s,top,x)

Step 1: [Check for stack overflow]  
If top>=n  
Then printf(“stack overflow”)  
Return

Step 2: [Increment Top]

Top<-top+1

Step 3: [ Insert element]

S[top]<-x

Step 4:[finished]

Return

**ALGORITHM FOR DELETING AN ELEMENT FROM A STACK:**

Function POP(s,top)

Step 1: [Check for stack underflow]  
If top=0  
Then printf(“stack underflow”)  
Exit

Step 2: [Decrement Top]

Top<-top-1

Step 3: [Return former top element of stackwwwww]

Return(S[top+1])

Step 4:[finished]

Return

# include <stdio.h>  
# define size 4  
int choice,top=0,a[size],item;  
main()  
{clrscr();

while(1)

{

printf(" \*\*\* MENU \*\*\*\n 1. PUSH\n 2. POP\n 3.

TRAVERSE\n 4. EXIT\n");

printf("enter your choice from menu:");  
scanf("%d",&choice);  
switch(choice)  
{

case 1:push();

break;

case 2:pop();

break;

case 3:traverse();

break;  
case 4:exit();  
default:printf("wrong choice\n");

}

}getch();

}push()

{  
if(size==top)  
printf("\*\*\* stack is full \*\*\*\n");  
else  
{

printf("enter the item to be pushed into the stack:");  
scanf("%d",&item);  
top++;  
a[top]=item;

}

}pop()

{  
if(top==0)  
printf("\*\*\* stack is empty \*\*\*\n");  
else  
{

# include <stdio.h>  
# define size 4  
int choice,top=0,a[size],item;  
main()  
{clrscr();

while(1)

{

printf(" \*\*\* MENU \*\*\*\n 1. PUSH\n 2. POP\n 3.

TRAVERSE\n 4. EXIT\n");

printf("enter your choice from menu:");  
scanf("%d",&choice);  
switch(choice)  
{

case 1:push();

break;

case 2:pop();

break;

case 3:traverse();

break;  
case 4:exit();  
default:printf("wrong choice\n");

}

}getch();

}push()

{  
if(size==top)  
printf("\*\*\* stack is full \*\*\*\n");  
else  
{

printf("enter the item to be pushed into the stack:");  
scanf("%d",&item);  
top++;  
a[top]=item;

}

}pop()

{  
if(top==0)  
printf("\*\*\* stack is empty \*\*\*\n");  
else  
{

148

item=a[top];  
top--;  
printf("the deleted item from stack is %d\n",item);

}

}traverse()

{  
int i;  
if(top==0)  
printf("\*\*\*\* stack is empty\*\*\*\*");  
else

{  
printf("\*\*\* stack display \*\*\*\n");  
for(i=1;i<=top;i++)  
if(i==top)  
printf("%d at %d ->top\n",a[i],i);  
else  
printf("%d at %d\n",a[i],i);

}

}

Input/Output:

\*\*\* MENU \*\*\*  
1. PUSH  
2. POP  
3. TRAVERSE

4. EXIT  
enter your choice from menu:1  
enter the item to be pushed into the stack:11  
\*\*\* MENU \*\*\*  
1. PUSH  
2. POP  
3. TRAVERSE  
4. EXIT  
enter your choice from menu:1  
enter the item to be pushed into the stack:12  
\*\*\* MENU \*\*\*  
1. PUSH  
2. POP  
3. TRAVERSE  
4. EXIT  
enter your choice from menu:1  
enter the item to be pushed into the stack:13  
\*\*\* MENU \*\*\*  
1. PUSH

2. POP

3. TRAVERSE

4. EXIT  
enter your choice from menu:1  
enter the item to be pushed into the stack:14  
\*\*\* MENU \*\*\*  
1. PUSH  
2. POP  
3. TRAVERSE  
4. EXIT  
enter your choice from menu:1  
\*\*\* stack is full\*\*\*

\*\*\* MENU \*\*\*  
1. PUSH  
2. POP  
3. TRAVERSE

4. EXIT  
enter your choice from menu:3  
\*\*\* stack display \*\*\*  
11 at 1  
12 at 2  
13 at 3  
14 at 4 ->top

\*\*\* MENU \*\*\*  
1. PUSH  
2. POP  
3. TRAVERSE

4. EXIT  
enter your choice from menu:2  
the deleted item from stack is 14  
\*\*\* MENU \*\*\*  
1. PUSH  
2. POP  
3. TRAVERSE  
4. EXIT  
enter your choice from menu:2  
the deleted item from stack is 13  
\*\*\* MENU \*\*\*  
1. PUSH  
2. POP  
3. TRAVERSE  
4. EXIT  
enter your choice from menu:2  
the deleted item from stack is 12

1. **Write a program for showing Queue operations using linked list.**

**Algorithm:**

Step 1: Start  
Step 2: define structure for queue  
Step 3: read choice  
Step 4: if choice = insert

i) read the element

ii) create a data structure

iii) if empty queue then front of queue pinter points to newly created data structure

iv) otherwise end of the queue points to newly created data structure

Step 5: if choice= remove

i) check if queue is empty . if so, print queue is empty

ii) otherwise read the element pointed by front of the queue temp pointer

points to front of queue

iii)front of queue points to next element

iv)free the element pointed by temp pointer

v) return the element

vi)print the element

Step 6: if choice = display

i) check of empty queue if so, print queue empty

ii)otherwise print the elements from front of the queue until the end of the

queue

step 7: if choice=exits stop

**program:**

#define true 1

#define false 0

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

struct q\_point

{

int ele;

struct q\_point\* n;

};

struct q\_point \*f\_ptr = NULL;

int e\_que(void);  
void add\_ele(int);  
int rem\_ele(void);  
void show\_ele();

/\*main function\*/  
void main()  
{

A

If  
empty  
queue

Print empty

queue

B

If ptr!

=NULL

Print ptr ->

ele

Ptr= ptr-> n

T

F

T

168

int ele,choice,j;

while(1)

{  
clrscr();  
printf("\n\n\*\*\*\*IMPLEMENTATION OF QUEUE USING

POINTERS\*\*\*\*\n");

printf("==============================================

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

printf("==============================================

");printf("\n\t[1] To insert an element");

printf("\n\t[2] To remove an element");  
printf("\n\t[3] To display all the elements");  
printf("\n\t[4] Exit");  
printf("\n\n\tEnter your choice:");  
scanf("%d", &choice);

switch(choice)

{

case 1:

{  
printf("\n\tElement to be inserted:");  
scanf("%d",&ele);  
add\_ele(ele);  
getch();  
break;

}

case 2:{

if(!e\_que())

{

j=rem\_ele();  
printf("\n\t%d is removed from the queue",j);  
getch();  
}else

{  
printf("\n\tQueue is Empty.");  
getch();

}break;

}

case 3:

program:

#define true 1

#define false 0

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

struct q\_point

{

int ele;

struct q\_point\* n;

};

struct q\_point \*f\_ptr = NULL;

int e\_que(void);  
void add\_ele(int);  
int rem\_ele(void);  
void show\_ele();

/\*main function\*/  
void main()  
{

A

If  
empty  
queue

Print empty

queue

B

If ptr!

=NULL

Print ptr ->

ele

Ptr= ptr-> n

T

F

T

168

int ele,choice,j;

while(1)

{  
clrscr();  
printf("\n\n\*\*\*\*IMPLEMENTATION OF QUEUE USING

POINTERS\*\*\*\*\n");

printf("==============================================

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

printf("==============================================

");printf("\n\t[1] To insert an element");

printf("\n\t[2] To remove an element");  
printf("\n\t[3] To display all the elements");  
printf("\n\t[4] Exit");  
printf("\n\n\tEnter your choice:");  
scanf("%d", &choice);

switch(choice)

{

case 1:

{  
printf("\n\tElement to be inserted:");  
scanf("%d",&ele);  
add\_ele(ele);  
getch();  
break;

}

case 2:{

if(!e\_que())

{

j=rem\_ele();  
printf("\n\t%d is removed from the queue",j);  
getch();  
}else

{  
printf("\n\tQueue is Empty.");  
getch();

}break;

}

case 3:

169

show\_ele();  
getch();  
break;

case 4:  
exit(1);  
break;

default:

printf("\n\tInvalid choice.");

getch();

break;

}

}

}/\* Function to check if the queue is empty\*/

int e\_que(void)

{

if(f\_ptr==NULL)  
return true;  
return false;

}/\* Function to add an element to the queue\*/

void add\_ele(int ele)

{  
struct q\_point \*queue = (struct q\_point\*)malloc(sizeof(struct q\_point));  
queue->ele = ele;  
queue->n = NULL;  
if(f\_ptr==NULL)  
f\_ptr = queue;

else

{

struct q\_point\* ptr;

ptr = f\_ptr;

for(ptr=f\_ptr ;ptr->n!=NULL; ptr=ptr->n);

ptr->n = queue;

}

}/\* Function to remove an element from the queue\*/

int rem\_ele()

{

struct q\_point\* queue=NULL;

if(e\_que()==false)

{  
int j = f\_ptr->ele;  
queue=f\_ptr;  
f\_ptr = f\_ptr->n;  
free (queue);

return j;

}else

{  
printf("\n\tQueue is empty.");  
return -9999;

}

}/\* Function to display the queue\*/

void show\_ele()

{

struct q\_point \*ptr=NULL;  
ptr=f\_ptr;  
if(e\_que())  
{

printf("\n\tQUEUE is Empty.");

return;

}

else

{

printf("\n\tElements present in Queue are:\n\t");

while(ptr!=NULL)

{  
printf("%d\t",ptr->ele);  
ptr=ptr->n;

}

}

}

Output:

\*\*\*\*IMPLEMENTATION OF QUEUE USING POINTERS\*\*\*\*

==============================================

MENU

==============================================  
[1] To insert an element  
[2] To remove an element  
[3] To display all the elements

[4] Exit  
Enter your choice:1  
Element to be inserted:23

1. Write a program to convert infix expression to postfix expression using stack.

ALGORITHM:

Step 1. start

Step 2. first initialize the stack to be empty

Step 3. for each character in the input string  
If input string is an operand, append to the output  
if the input string is a left paranthesis , push it onto the stack  
else  
if stack is empty or the operator has higher priority than the operator on

the topof stack or

the top of the stack is opening parenthesis

then  
push the operator onto the stack  
else  
pop the operator from the stack and append to the output

Step 4. if the input string is a closing parenthesis , pop operators from the stack  
and append the operators  
to the output until an opening parenthesis is encountered. Pop the  
opening parenthesis from the stack  
and discard it.

Step 5. if the end of the input string is encountered , then iterate the loop until the  
stack is not empty. Pop  
the stack and append the remaining input string to the output.

Step 6. stop

Program:

#include<stdio.h>  
#include<ctype.h>  
#include<string.h>  
static char str[20];  
int top=-1;  
main()  
{char in[20],post[20],ch;

int i,j,l;  
clrscr();  
printf("enter the string");  
gets(in);  
l=strlen(in);  
for(i=0,j=0;i<l;i++)  
if(isalpha(in[i]))  
post[j++]=in[i];  
else  
{if(in[i]=='(')

push(in[i]);  
else if(in[i]==')')  
while((ch=pop())!='(')  
post[j++]=ch;  
else  
{while(priority(in[i])<=priority(str[top]))

post[j++]=pop();  
push(in[i]);  
}}while(top!=-1)

post[j++]=pop();  
post[j]='\0';  
printf("\n equivalent infix to postfix is:%s",post);  
getch();  
}priority (char c)

{switch(c)

{case'+':

case'-': return 1;

case'\*':

case'/':

return 2;

case'$':return 3;  
}return 0;  
}push(char c)  
{str[++top]=c;  
}pop()  
{return(str[top--]);  
}

Input/Output:

enter the string(a+b)-(c-d)\*e/f

equivalent infix to postfix is:ab+cd-e\*f/-

enter the stringa+b/c\*d

equivalent infix to postfix is:abc/d\*+

1. Write a program for evaluating postfix expression.

**ALGORITHM:**

Step 1: Start

Step 2: Assign top=-1  
Step 3: Read the input expression  
Step 4: for i=0;s[i]!=’\0’ in steps of 1

Step 5: If isdigit(ch)  
Step 5.1:Push(ch)  
Step 6: otherwise  
Step 6.1:op1=pop()  
Step 6.2: op2=pop()  
Step 7: c=op2+op1  
Step 8: Push(c)  
Step 9: c=op2-op1  
Step 10: Push(c)  
Step 11: c=pow(op2,op1)  
Step 12: Push(c)  
Step 13: c=op2/op1  
Step 14:Push(c)  
Step 15: Print the result  
Step 16:Push(int x)  
Step 17:Increment top by 1  
Step 18: s1.item(s1.top3)=x  
Step 19:pop()  
Step 20: Read x  
Step 21: x1=s1.item[s1.top]  
Step 22:s1.top—  
Step 23:return x  
Step 24: Stop

Program:

#include<stdio.h>  
#include<ctype.h>  
int stk[10],top=0,op1,op2,i;  
main()  
{char postexp[10];

clrscr();  
printf("enter the postfix expression:");  
gets(postexp);  
for(i=0;postexp[i]!='\0';i++)  
{if(isdigit(postexp[i]))

push(postexp[i]-48);  
else  
{op1=pop();

op2=pop();  
switch(postexp[i])  
{case '+':push(op1+op2);

break;

case '-':push(op1-op2);

break;

case '\*':push(op1\*op2);

break;

case '/':push(op1/op2);

break;

case '%':push(op1%op2);

break;  
case '.':exit();  
}}}printf("the result of postfixexpression is: %d",pop());

getch();

}

pop()

{return(stk[top--]);

}

push(int x)

{top++;

stk[top]=x;

}

Input/Output:

enter the postfix expression:234+-

the result of postfix expression is: 5

1. Write a program for printing pre-order traversal of Binary Tree.

**For PREORDER function**Step 1: Start  
Step 2: If t!=null  
Step 3: Write data  
Step 4: Call postorder(t->lc)  
Step 5: Call postorder(t->rc)  
Step 6: Stop

1. Write a program for printing post-order traversal of Binary Tree.

**For POSTORDER function**

Step 1: Start  
Step 2: If t!=null  
Step 3: Call postorder(t->lc)  
Step 4: Call postorder(t->rc)  
Step 5: Write data  
Step 6: Stop

1. Write a program for printing in-order traversal of Binary Tree.

**For INORDER function**

Step 1: Start

Step 2: If t!=null  
Step 3: Call inorder(t->lc)  
Step 4: Write t->data  
Step 5: Call inorder(t->rc)  
Step 6: Stop

Program

#include<stdio.h>  
#include<alloc.h>  
struct bstnode  
{int data;

struct bstnode \*lc,\*rc;  
}\*root,\*a[20],\*b[20];  
int top=-1,top1=-1,n,i;  
main()  
{int ch,ele;

struct bstnode \*t,\*insert(),\*pop();  
clrscr();  
t=root=NULL;  
while(1)  
{printf("\n\*\*\*\* M E N U\*\*\*\* \n");

printf("1.INSERT\n2.RECURSSIVE TRAVERSE\n3.NON-RECURSIVE  
TRAVERSE\n4.EXIT\n");  
printf("Enter your choice: ");  
scanf("%d",&ch);  
switch(ch)  
{case 1: printf("Enter how many elements u want to insert:");

scanf("%d",&n);  
printf("Enter tree elements: ");  
for(i=1;i<=n;i++)  
{scanf("%d",&ele);

t=insert(t,ele);

}break;

case 2: /\* RECURSSIVE TRAVERSE \*/  
if(t==NULL)  
printf("\*\*\*\* TREE IS EMPTY\*\*\*\*");  
else  
{printf("INORDER :");

inorder(t);

printf("\nPREORDER :");

preorder(t);  
printf("\nPOSTORDER :");  
postorder(t);  
}break;

case 3: /\* NON-RECURSSIVE TRAVERSE \*/  
if(t==NULL)  
printf("TREE IS EMPTY");  
else  
{

printf("INORDER :");  
nrinorder(t);  
printf("\nPREORDER :");  
nrpreorder(t);  
printf("\nPOSTORDER :");  
nrpostorder(t);  
}break;

case 4:

exit();

}}}struct bstnode \*insert(struct bstnode \*x,int y)

{  
if(x==NULL)  
{x=malloc(sizeof(struct bstnode));

x->data=y;  
x->lc=NULL;  
x->rc=NULL;  
}else

{if(y<x->data)

x->lc=insert(x->lc,y);  
else  
x->rc=insert(x->rc,y);  
return(x);  
}

}inorder(struct bstnode \*x)

{  
if(x!=NULL)  
{inorder(x->lc);

printf("%3d",x->data);  
inorder(x->rc);  
}

}preorder(struct bstnode \*x)

190

{  
if(x!=NULL)  
{printf("%3d",x->data);

preorder(x->lc); preorder(x->rc); }

}postorder(struct bstnode \*x)

{  
if(x!=NULL)  
{postorder(x->lc);

postorder(x->rc);  
printf("%3d",x->data);  
}

}nrinorder(struct bstnode \*x)  
{struct bstnode \*l;  
l=x;

do

{  
while(l!=NULL)  
{push(l);

l=l->lc;

}while(top>-1)

{  
l=pop();  
printf("%d",l->data);  
if(l->rc!=NULL)  
{l=l->rc;

break;

}else

l=NULL;

}  
}while(l!=NULL);  
}nrpreorder(struct bstnode \*x)

{struct bstnode \*l;

l=x;

do

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

if(l->rc!=NULL)  
push(l->rc);  
l=l->lc;  
if(l==NULL&&top>-1)  
l=pop();  
}while(l!=NULL);

}nrpostorder(struct bstnode \*x)

{  
struct bstnode \*l;  
l=x;  
do  
{

while(l!=NULL)

{  
push(l);  
if(l->rc!=NULL)  
{push(l->rc);

b[++top1]=l->rc;

}l=l->lc;  
}do  
{l=pop();  
if(l!=b[top1])

printf("%3d",l->data);  
else  
{top1-=1;

break;

}} while(top>-1);

}while(l!=NULL&&top>-1);

}push(struct bstnode \*y)  
{top+=1;  
a[top]=y;

}struct bstnode \*pop()

{

192

000000

return a[top--];

}

Input/Output:

Enter your choice

1.Insert 2.Delete 3.Traversal

Enter the element 92  
Enter your choice  
1. Insert 2.Delete 3. Traversal  
Enter the element 26  
Enter your choice

1.Insert 2.Delete 3.Traversal

Enter the element 12

Enter your choice

1.Insert 2.Delete 3.Traversal

Enter the element 123

Enter your choice

1.Insert 2.Delete 3.Traversal

Enter the element 135

Enter your choice

1.Insert 2.Delete 3.Traversal

Enter the element 128

Enter your choice

1.Insert 2.Delete 3.Traversal

3InorderSequence: 12 26 92 123 128 135

Preorder sequence:92 26 12 123 135 128

Postorder sequence: 12 26 128 135 12 92

1. Write a program to implement doubly linked list.

(Create , insert, delete, & traverse)

ALGORITHM :

Step 1: Start  
Step 2: Declare a structure with \*next, \*pre  
Step 3: Declare \*start, \*new ,\*l as structure pointers  
Step 4: Print main menu  
Step 5: Read choice  
Step 6: Switch choice

6.1: call insert function if choice==1

6.2: call delete function if choice==2

6.3: call view function if choice==3  
Step 7: Stop  
Step 8: Start of insert function  
Step 9: Read e  
Step 10: If start==null  
Step 11: Create a new node  
Step 12: Start->data=e  
Step 13: Start->next=null  
Step 14: Start->pre=null  
Step 15: read choice, where to insert  
Step 16: if choice==1  
Step 16.1: Create a new mode  
Step 16.2: new -> data=e

129

Step 16.3: new -> next=start

Step 16.4: start->pre=new  
Step 16.5: new->pre=null  
Step 16.6: Start->new

Step 17: otherwise if choice==2  
17.1: read position p  
17.2: l=start  
17.3: while i<(p-1)  
17.4: incrent i  
17.5: l=l->next  
17.6: new -> data =e  
17.7: new -> pre=l  
17.8: new->next=new  
17.9: l-> next=new  
17.10: l->next->pre=new

Step 18: if choice==3  
18.1: l=start  
18.2: while l->next!=null  
18.3: l=l->next  
18.4: create a new mode  
18.5: new->data=e  
18.6: new->next=null  
18.7: l->next=new

130

18.8: new->pre=l  
Step19: end of insert function  
Step20: start of deletion function  
Step21: write menu  
Step22: read choice  
Step23: if choice==1

23.1: temp=start->data  
23.2: start=start->next  
23.3: start->pre=null

Step24: if choice==2  
24.1: read position  
24.2: l=start  
24.3: while (i=1 <p-1)  
24.4: l=l->next  
24.5: increment I by 1  
24.6: temp=l-next->data  
24.7: l->next=l->next->next  
24.8: l->next->pre=l

Step25: if choice==3  
25.1: read l=start  
25.2: while l->next->next!= null  
25.3: l=l->next  
25.4: temp=l->next->data

25.5: l->next=null  
Step26: end of delete function  
Step27: start of view function  
Step28: read choice  
Step29: if choice==1

29.1: l=next  
29.2: while (l->next!= null)  
29.3: write l-> data, l=l->next  
29.4: write l->data

Step30: if choice==2  
30.1: l=start  
30.2: while l!=start  
30.3: write l->data  
30.4: l=l->pre  
30.5: write l->data

Step31: end of function view

Program:

133

#include<stdio.h>

#include<malloc.h>  
/\* START OF STRUCTURE DEFINITION \*/  
struct link

{  
int data;  
struct link \*next;  
struct link \*prev;

}\*start,\*new,\*temp,\*l,\*l1,\*t,\*start1;  
/\* END OF STRUCTURE DEFINITION \*/  
int item,ch,i,j,p,n;

/\* VARIABLE DECLARATION \*/  
/\* START OF MAIN FUNCTION \*/  
main()

{  
start=NULL;  
start1=NULL;  
clrscr();  
printf("\*\*\*\* MENU\*\*\*\*");  
printf("\n1.Insertion\n2.Deletion\n3.Traverse\n4.search\n5.sort\n6.merge\n  
7.reverse\n8.exit\n");  
while(1)  
{

printf("enter your choice:");  
scanf("%d",&ch);  
switch(ch)  
{

case 1:insert();

break;

case 2:delete();

break;

case 3:display();

break;

case 4:search();

break;

case 5:sort();

break;

134

case 6:merge();

break;

case 7:reverse();

break;  
case 8:exit();  
}

}  
getch();  
}/\* END OF MAIN FUNCTION \*/

/\* START OF INSERT FUNCTION \*/

insert()

{  
l=start;  
printf("enter an item to be inserted:");  
scanf("%d",&item);  
new=malloc(sizeof(struct link));  
new->data=item;  
if(start==NULL)  
{

new->prev=NULL;  
new->next=NULL;  
start=new;

}

else

{  
printf("1.start\n2.middle\n3.end\n");  
printf("enter the place to insert item:");  
scanf("%d",&ch);  
if(ch==1)  
{

new->next=start;  
new->prev=NULL;  
start=new;

}if(ch==2)

{  
printf("enter the position to place item:");  
scanf("%d",&p);  
for(i=1;i<p-1;i++)  
l=l->next;  
new->prev=l;

new->next=l->next;

l->next=new;

}if(ch==3)

{  
while(l->next!=NULL)  
l=l->next;  
new->prev=l;  
new->next=NULL;  
l->next=new;

}

}

}/\* END OF INSERT FUNCTION \*/

/\* START OF DELETE FUNCTION \*/

delete()

{  
l=start;  
if(start==NULL)  
printf("\*\*\* LIST IS EMPTY \*\*\*");  
else  
{

printf("1.start\n2.middle\n3.end");  
printf("enter the place to delete the item:");  
scanf("%d",&ch);  
if(ch==1)  
{

item=start->data;  
printf("deleted item is :%d",item);  
start=start->next;  
start->prev=NULL;

}if(ch==2)

{  
printf("enter the position to delete an item:");  
scanf("%d",&p);  
if(l->next==NULL)  
{

item=l->data;  
printf("deleted item is:%d",item);  
l=start=NULL;

}else

136

{  
for(i=1;i<p-1;i++)  
l=l->next;  
item=l->next->data;  
printf("deleted item is:%d",item);  
l->next=l->next->next;  
l->next->prev=l;

}

}if(ch==3)

{

if(l->next==NULL)

{  
item=l->data;  
printf("deleted item is :%d",item);  
l->prev=NULL;  
l=start=NULL;

}else

{  
while(l->next->next!=NULL)  
l=l->next;  
item=l->next->data;  
printf("deleted item is:%d",item);  
l->next=NULL;

}

}

}

}/\* END OF DELETE FUNCTION \*/

/\* START OF DISPLAY FUNCTION \*/

display()

{  
if(start==NULL)  
printf("\*\*\* LIST IS EMPTY \*\*\*\n");  
else

{  
for(l=start;l->next!=NULL;l=l->next)  
if(l==start)  
printf("\nstart:%d",l->data);  
else  
printf("\n %8d",l->data);  
if(l->next==NULL)

printf("\n last:%d",l->data);

137

}

}/\* END OF DISPLAY FUNCTION \*/

/\* START OF SEARCH FUNCTION \*/

search()

{  
int f=0;  
if(start==NULL)  
printf(" \*\*\* LIST IS EMPTY \*\*\* ");  
else  
{

printf("enter the search item:");  
scanf("%d",&item);  
for(l=start,i=1;l!=NULL;l=l->next,i++)  
if(item==l->data)  
{

f=1;

break;

}if(f==1)

printf("item %d found at position %d",item,i);  
else  
printf("item %d not found in list",item);

}

}/\* END OF SEARCH FUNCTION \*/

/\* START OF SORT FUNCTION \*/

sort()

{  
int t;  
if(start==NULL)  
printf(" \*\*\* LIST IS EMPTY \*\*\* ");  
else  
{

for(l1=start;l1->next!=NULL;l1=l1->next)  
for(l=start;l->next!=NULL;l=l->next)  
if(l->data > l->next->data)  
{

t=l->next->data;  
l->next->data=l->data;  
l->data=t;

}printf("THE SORTED ORDER IS:");

for(l=start;l!=NULL;l=l->next)

printf("%3d",l->data);

}printf("\n");

}/\* END OF SORT FUNCTION \*/

/\* START OF MERGE FUNCTION \*/

merge()

{

printf("enter number items to be inserted in second list:");  
scanf("%d",&n);  
for(j=1;j<=n;j++)

{  
l1=start1;  
printf("enter an item:");  
scanf("%d",&item);  
new=malloc(sizeof(struct link));  
new->data=item;  
if(start1==NULL)  
{

new->prev=NULL; new->next=NULL; start1=new;

}

else

{  
printf("1.start\n2.middle\n3.end\n");  
printf("enter the place to insert item:");  
scanf("%d",&ch);  
if(ch==1)  
{

new->next=start1;  
new->prev=NULL;  
start1=new;

}if(ch==2)

{  
printf("enter the position to place item:");  
scanf("%d",&p);  
for(i=1;i<p-1;i++)

139

l1=l1->next;  
new->prev=l1;  
new->next=l1->next;  
l1->next=new;

}if(ch==3)

{  
while(l1->next!=NULL)  
l1=l1->next;  
new->prev=l1;  
new->next=NULL;  
l1->next=new;

}

}

}if(start==NULL)

start=start1;

else

{  
l=start;  
while(l->next!=NULL)  
l=l->next;  
for(l1=start1;l1->next!=NULL;l1=l1->next)  
{

l->next=l1;

l=l->next;

}

}printf(" \*\*\* LIST IS MERGED \*\*\* \n");

}/\* END OF MERGE FUNCTION \*/

/\* START OF REVERSE FUNCTION \*/

reverse()

{  
if(start==NULL)  
printf(" \*\*\* LIST IS EMPTY \*\*\*\n ");  
else

{  
l=start;  
l1=t=NULL;  
while(l!=NULL)  
{

l1=t;

140

t=l;  
l=l->next;  
t->next=l1;

}start=t;

printf(" \*\*\* LIST IS REVERSED \*\*\* \n");

}

}/\* END OF REVERSE FUNCTION \*/

Input/Output:

\*\*\*\*M EN U \*\*\*\*  
1.Insertion  
2.Deletion  
3.Traverse  
4.search  
5.sort  
6.merge  
7.reverse  
8.exit  
enter your choice:1  
enter an item to be inserted:10  
enter your choice:1  
enter an item to be inserted:20  
1.start  
2.middle  
3.end  
enter the place to insert item:1  
enter your choice:1  
enter an item to be inserted:30  
1.start  
2.middle  
3.end  
enter the place to insert item:3  
enter your choice:1  
enter an item to be inserted:40  
1.start  
2.middle  
3.end

000

25.5: l->next=null  
Step26: end of delete function  
Step27: start of view function  
Step28: read choice  
Step29: if choice==1

29.1: l=next  
29.2: while (l->next!= null)  
29.3: write l-> data, l=l->next  
29.4: write l->data

Step30: if choice==2  
30.1: l=start  
30.2: while l!=start  
30.3: write l->data  
30.4: l=l->pre  
30.5: write l->data

Step31: end of function view

1. **Write a program for implementing Bubble Sort.**

Algorithm:

i)Bubble Sort:

1. start  
2. read the value of n  
3. for i= 1 to n increment in steps of 1

Read the value of ith element into array  
4. call function to sort (bubble\_sort(a,n))  
5. for i= 1 to n increment in steps of 1

print the value of ith element in the array

6. stop

BUBBLE SORT FUNCTION

1. start  
2. initialise last to n  
3. for i= 1 to n increment in steps of 1

begin

4. initialise ex to 0

5. for i= 1 to last-1 increment in steps of 1

begin

6. if k[i]>k[i+1] goto step 7 otherwise goto step 5

begin

7. assign k[i] to temp  
assign k[i+1] to k[i]  
assign temp to k[i+1]  
increment ex by 1

end-if

end inner for loop

8. if ex!=0

assign last-1 to last

end for loop

9. stop

Program:

#include<stdio.h>  
main()  
{int i,j,t,a[5],n;

clrscr();  
printf("enter the range of array:");  
scanf("%d",&n);  
printf("enter elements into array:");  
for(i=0;i<n;i++)  
scanf("%d",&a[i]);  
for(i=0;i<n-1;i++)  
for(j=i+1;j<n;j++)  
if(a[i]>a[j])  
{t=a[i];

a[i]=a[j];  
a[j]=t;  
}printf("the sorted order is:");

for(i=0;i<n;i++)  
printf("\t%d",a[i]);  
getch();

}Input/Output:

enter the range of array:3  
enter elements into array:3  
21the sorted order is: 1

2

3

1. **Write a program for implementing Heap sort.**

Heap sort

SWAP FUNCTION  
1. start  
2. assign \*a to temp  
3. assign \*b to \*a  
4. assign temp to \*b  
5. stop

HEAP SORT  
1. start  
2. assign n to i and a[n] to item  
3. if i > 1 and a[i/2]< item repeat through step 4 other wise goto

step 5

begin

4.

assign a[i/2] to a[i] and i/2 to i

end if  
5. assign item to a[i]  
6. stop

Program:

#include<stdio.h>  
int a[20];  
main()  
{int n,i;

clrscr();  
printf("Enter number of elements: ");  
scanf("%d",&n);  
printf("Enter %d elements: ",n);  
for(i=1;i<=n;i++)  
scanf("%d",&a[i]);  
heapsort(n);  
printf("Sorted elements are: \n");  
for(i=1;i<=n;i++)  
printf("%3d",a[i]);  
getch();  
}heapsort(int n)

{int t;

while(n>1)

{  
maxheap(n);  
t=a[1];  
a[1]=a[n];  
a[n]=t;  
n=n-1;

}}maxheap(int n)

{int i,t,j;

for(i=2;i<=n;i++)

{  
t=a[i];  
j=i;  
while(a[j/2]<t&&j>1)  
{a[j]=a[j/2];

j=j/2;

}a[j]=t;

}}

Input/Output:

Enter number of elements: 4  
Enter 4 elements: 23  
412

8Sorted elements are:

4 8 12 23

1. **Write a program for implementing Quick sort.**

Algorithm:

Quick Sort:

1. start

2. if lowerbound < upperbound repeat through steps 3 to 13 otherwise

goto step 14

begin  
3. assign lowerbound to i,upperbound to j, i to pivot  
4. if i<j repeat through steps 5 to 10 otherwise goto step \_Begin

5. if a[i]<=k[pivot] and i< upperbound repeat through step 6 otherwise

goto step 7

begin

6. assign i+1 to i

end if

7. if k[j] > k[pivot] repeat through step 8 otherwise goto step 9

begin

8. assign j-1 to j

end if

9. if i< j goto step 10 other wise goto step 4

Begin

10.

call function to swap k[i] and k[j]

end if

end if

11. call function to swap k[pivot] and k[j]

12. call function qsort(x,lowerbound,j-1)

13. call function qsort(x,j+1,upperbound)

end if

14. stop

#include<stdio.h>

main()

{  
int x[10],i,n;  
clrscr();  
printf("enter no of elements:");  
scanf("%d",&n);  
printf("enter %d elements:",n);  
for(i=1;i<=n;i++)  
scanf("%d",&x[i]);  
quicksort(x,1,n);  
printf("sorted elements are:");  
for(i=1;i<=n;i++)  
printf("%3d",x[i]);  
getch();

}quicksort(int x[10],int first,int last)

{  
int pivot,i,j,t;  
if(first<last)  
{

pivot=first;  
i=first;  
j=last;  
while(i<j)  
{

while(x[i]<=x[pivot] && i<last)  
i++;  
while(x[j]>x[pivot])

j--;

if(i<j)

{  
t=x[i];  
x[i]=x[j];  
x[j]=t;

}

}  
t=x[pivot];  
x[pivot]=x[j];  
x[j]=t;  
quicksort(x,first,j-1);  
quicksort(x,j+1,last);

}

}

\*\*\*\*\*OUTPUT \*\*\*\*\*

enter no of elements:6  
enter 6 elements:23  
12  
45  
34  
21  
87  
sorted elements are: 12 21 23 34 45 87

1. **Write a program for implementing Merge sort.**

Algorithm: main program

Step1: Start  
Step2: declare the merge sort function  
Step3: Declare the array and their size and initailaze the j=0  
Step4: read the array elements and then sort these elements.  
Step5: read the array elements before the merge sort and then display the

elements.  
Step6: call the merge sort function  
Step7: display the array elements after merge sort by using the following stament.

for( j=0;j<Max\_ary;j++)

Step8: Stop

**Subprogram**

Step1:initialize the array excuting[MAX\_ARY] and  
j=0,mid=0,mrg1=0,mrg2=0,size=start-end+1  
Step2: check the condition if(end==start) then return

Step3: calculate the mid value

Mid=(end+start)/2  
Step4: call themerge\_sort(x,end,mid)  
Step5:merge\_sort(x,mid+1,start)  
Step6: performing the looping operation

For(j=0;j<SIZE;j++) then its true

Executing[j]=x[end+1]  
Mrg1=0;  
Step7: calculate the mrg2=mid-end+1  
Step8: performing the looping operation

For(j=0;j<SIZE;j++) then its true then goto step9

Step9: check the condition

i) if(mrg2<=start-end) is true goto ii). If not goto Step12.

ii) If(mrg1<=mid-end) is true goto iii). If not goto step11

iii) If(executing[mrg1]>executing[mrg2]) is true then follows.

If not goto step10.

X[j+end]= executing[mrg2++]  
Step10: x[j+end]=executing[mrg1++]. If not goto Step11  
Step11: x[j+end]= executing[mrg2++]  
Step12: x[j+end]=executing[mrg1++]  
Step13: return to main program

Program:

#include <stdio.h>

#include <stdlib.h>  
#define MAX\_ARY 10  
void merge\_sort(int x[], int end, int start);  
int main(void) {

int ary[MAX\_ARY];

int j = 0;

printf("\n\nEnter the elements to be sorted: \n");

for(j=0;j<MAX\_ARY;j++)

scanf("%d",&ary[j]);

/\* array before mergesort \*/

printf("Before :");

for(j = 0; j < MAX\_ARY; j++)

printf(" %d", ary[j]);  
printf("\n");  
merge\_sort(ary, 0, MAX\_ARY - 1);  
/\* array after mergesort \*/

printf("After Merge Sort :");

for(j = 0; j < MAX\_ARY; j++)

printf(" %d", ary[j]);

printf("\n");

getch();

}

/\* Method to implement Merge Sort\*/

void merge\_sort(int x[], int end, int start) {  
int j = 0;  
const int size = start - end + 1;  
int mid = 0;  
int mrg1 = 0;  
int mrg2 = 0;  
int executing[MAX\_ARY];

if(end == start)

return;

mid = (end + start) / 2;

merge\_sort(x, end, mid);

merge\_sort(x, mid + 1, start);

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

executing[j] = x[end + j];

mrg1 = 0;

mrg2 = mid - end + 1;

for(j = 0; j < size; j++) {

if(mrg2 <= start - end)

if(mrg1 <= mid - end)  
if(executing[mrg1] > executing[mrg2])  
x[j + end] = executing[mrg2++];  
else  
x[j + end] = executing[mrg1++];

else

x[j + end] = executing[mrg2++];  
else  
x[j + end] = executing[mrg1++];

}

}Output:

Enter the elements to be sorted:

8 2 3 4 1 5 7 6 9 0

Before : 8 2 3 4 1 5 7 6 9 0

After Merge Sort : 0 1 2 3 4 5 6 7 8 9

1. **Write a program for implementing Insertion sort.**

**Algorithm:**

**Insertion Sort:**1. start  
2. for i= 1 to n increment in steps of 1

begin

assign k[i] to temp

3. forj=i-1 down to j>=0 and temp<k[j]  
begin  
assign k[j] to k[j+1]

end inner for loop

4. assign temp to k[j+1]

end for loop

5. stop

#include<stdio.h>  
main()  
{

int i,j,t,a[10],n,p=0;  
clrscr();  
printf("enter the range of array:");  
scanf("%d",&n);  
printf("enter elements into array:");  
for(i=0;i<n;i++)  
scanf("%d",&a[i]);  
for(i=1;i<n;i++)  
{

t=a[i];  
for(p=i;p>0 && a[p-1]>t;p--)  
a[p]=a[p-1];  
a[p]=t;

}printf("the sorted order is:");

for(i=0;i<n;i++)  
printf("\t%d",a[i]);  
getch();

}\*\*\*\*\*OUTPUT \*\*\*\*\*

enter the range of array:5  
enter elements into array:5  
4321the sorted order is: 1

2

3

4

5

enter the range of array:6  
enter elements into array:23  
12  
89  
45  
67  
34  
the sorted order is: 12

23

34

45

67

89

Objective 47:

Write a c program for selectioon sort .

Algo

Algorithm:

1) Start

2) Initiliaze the variables I,j,temp and arr[]

3) Read the loop and check the condition. If the condition is true  
print the array elements and increment the I value. Else goto step 4

4) Read the loop and check the condition. If the condition true then goto next loop.

5) Read the loop and check the condition. If the condition true then goto if condition

6) If the condition if(arr[i]>arr[j]) is true then do the following steps

i) temp=arr[i]

ii) arr[i]=arr[j]

iii) arr[j]=temp  
7) increment the j value  
8) perform the loop operation for the displaying the sorted

elements.

9) print the sorted elements

10) stop

**Program:**

#incude<stdio.h>  
#incude<conio.h>  
Void main()  
{

Int arr[5]={25,17,31,13,2};  
Int I,j,temp;  
Clrscr();  
Printf(“selection sort\n”);  
Printf(“\n array before sorting:\n”);  
For(i=0;i<=3;i++)  
Printf(“%d\t,arr[i]”);  
For(i=0;i<=3;i++)  
{

For(j=j+1;j<=4;j++)

{

If(arr[i]>arr[j])

{  
Temp=arr[i];  
Arr[i]=arr[j];

Arr[j]=temp;

}

}

}Printf(“\n\n array after sortong:\n”);

For(i=0;i<=4;i++)  
Printf(“%d\t”,arr[i]);  
Getch();  
}Sampe input & output:

1) Section sort

Array before sorting:

25 17 31

13 2

Array after sorting:

2 13 17 25 31

Thank you,