**ASSIGNMENT - 1**

**Problem Statement**

Write a menu driven program to implement an array with different type of operations: Insertion, Deletion, Traverse, Searching.

Insertion and Deletion have different way of execution enlisted following:

1. At the beginning
2. At the end
3. At any given position
4. After an element
5. Before an element

**Solution**

**Theory**

Here we use Array as the Data Structure. We have an array named **arr** and its max size is **N** and **len** is the current size of the array.

**Algorithm**

create\_array()

Begin

while true then

do

print “Enter the size =”

input len

if len > N or len <1

then

print “Wrong size”

else

break

end if

end while

print “Enter the elements”

i 🡨 0

while i < len then

do

input arr[i]

i 🡨 i+1

end while

End

Insert\_beg()

Begin

if len >= N

then

print “array is full”

return

end if

print “Enter the element”

input n

i 🡨 0

while i < len then

do

temp 🡨 arr[i]

arr[i] 🡨 n

n 🡨 temp

i 🡨 i+1

end while

arr[i] 🡨 n

len 🡨 len + 1

End

insert\_end()

Begin

if len>= N

then

print “array is full”

return

end if

print “enter the element”

input n

arr[len] 🡨 n

len 🡨 len + 1

End

Insert\_any()

Begin

if len>= N

then

print “array is full”

return

end if

print “the element”

input n

print “Position”

input pos

i 🡨 pos -1

while i <len then

do

temp 🡨 arr[i]

arr[i] 🡨 n

n 🡨 temp

i 🡨 i+1

end while

if i >= len

then

arr[len] 🡨 n

else

arr[i] 🡨 n

end if

len 🡨 len +1

End

insert\_after()

Begin

if len>= N

then

print “array is full”

return

end if

print “the element”

input n

print “Enter the element after want to insert”

input af

i🡨 0

while i<len then

do

if arr[i] = af

then

break

end if

end while

if i=len

then

print “element not exist”

return

end if

i🡨i+1

while i<len then

do

temp 🡨arr[i]

arr[i]🡨n

n🡨temp

i🡨i+1

end while

arr[i] 🡨 n

len 🡨 len+1

End

insert\_before()

Begin

if len>= N

then

print “array is full”

return

end if

print “the element”

input n

print “enter the element before want to insert”

input bf

i🡨 0

while i<len then

do

if arr[i] = af

then

break

end if

end while

if i=len

then

print “element not exist”

return

end if

while i<len then

do

temp 🡨arr[i]

arr[i]🡨n

n🡨temp

i🡨i+1

end while

arr[i] 🡨 n

len🡨len+1

End

del\_beg()

Begin

if len = 0

then

print “arry is empty”

return

end if

print arr[0] is deleted

i🡨0

while i<len-1 then

do

arr[i]🡨arr[i+1]

i🡨i+1

end while

len 🡨 len -1

End

del\_end()

Begin

if len = 0

then

print “arry is empty”

return

end if

print arr[len-1] deleted

len🡨len-1

End

del\_any()

Begin

if len = 0

then

print “arry is empty”

return

end if

print “enter the element to delete”

input n

search the positon of that element in the “arr”

if element not found in the array then return

else

store the location in the i

print arr[i] is deleted

while i<len then

do

arr[i]🡨arr[i+1]

i🡨i+1

end while

end if

len 🡨 len – 1

End

del\_after()

Begin

if len = 0

then

print “arry is empty”

return

end if

print “Enter the element whose after element want to delete”

input n

if n is not in the “arr” then return and print “not exist that element”

else

if found at the last position then also return and print “not possible”

else

store the location into i

print arr[i+1] is deleted

i🡨i+1

while i<len-1 then

do

arr[i]🡨arr[i+1]

i🡨i+1

end while

end if

end if

len🡨len-1

End

del\_before()

Begin

if len = 0

then

print “arry is empty”

return

end if

print “Enter the element whose before element want to delete”

input n

if n is not in the “arr” then return and print “not exist that element”

else

if found at the first position of ‘arr’ then return and print “not Possible”

else

store the location into i

print arr[i-1] is deleted

i🡨i-1

while i<len-1 then

do

arr[i]🡨arr[i+1]

i🡨i+1

end while

end if

end if

len🡨len-1

End

display()

Begin

print “elements are”

i🡨0

while i<len then

do

print arr[i]

i🡨i+1

end while

End

function “main()”

Begin

choice one of the operation to perform on the ‘arr’

1. create\_array()
2. insert\_beg()
3. insert\_end()
4. insert\_any()
5. insert\_after()
6. insert\_before()
7. del\_beg()
8. del\_end()
9. del\_any()
10. del\_after()
11. del\_before()
12. display()

End

**Source Code**

#include <stdio.h>

#include <stdlib.h>

#define N 50 // Hight Number of element can store

int arr[N]; // Array name

int len; // Array size

// Create a array

void create\_array()

{

int i;

while(1)

{

printf("\nEnter the Number Of Elements (less then %d)= ",N);

scanf("%d",&len);

if(len>N || len<1)

printf("Wrong size of the Array...");

else

break;

}

printf("Enter the Elements :-\n");

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

{

printf("A[%d] = ",i+1 );

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

}

}

// Insert an element at beg

void insert\_beg()

{

int i,n,temp;

if(len>=N)

{

printf("\nArray is full...");

return;

}

printf("\nEnter the element = ");

scanf("%d",&n);

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

{

temp=arr[i];

arr[i]=n;

n=temp;

}

arr[i]=n;

len=len+1;

}

// Insert an element at end

void insert\_end()

{

int n;

if(len>=N)

{

printf("\nArray is full...");

return;

}

printf("\nEnter the element = ");

scanf("%d",&n);

arr[len]=n;

len=len+1;

}

// Insert an element at any Position

void insert\_any()

{

int i,n,temp,pos;

if(len>=N)

{

printf("\nArray is full...");

return;

}

printf("\nEnter the element = ");

scanf("%d",&n);

printf("\nEnter the Position =");

scanf("%d",&pos);

for(i=pos-1;i<len;i++)

{

temp=arr[i];

arr[i]=n;

n=temp;

}

if(i>=len)

arr[len]=n;

else

arr[i]=n;

len=len+1;

}

// Insert an element after an element

void insert\_after()

{

int i,n,af,temp;

if(len>=N)

{

printf("\nArray is full...");

return;

}

printf("\nEnter the element to insert = ");

scanf("%d",&n);

printf("\nEnter the element after you want to insert %d = ",n);

scanf("%d",&af);

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

if(arr[i]==af)

break;

if(i==len)

{

printf("\n%d is not exist in the array",af);

return;

}

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

{

temp=arr[i];

arr[i]=n;

n=temp;

}

arr[i]=n;

len=len+1;

}

// Insert an element before an element

void insert\_before()

{

int i,n,bf,temp;

if(len>=N)

{

printf("\nArray is full...");

return;

}

printf("\nEnter the element to insert = ");

scanf("%d",&n);

printf("\nEnter the element before you want to insert %d = ",n);

scanf("%d",&bf);

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

if(arr[i]==bf)

break;

if(i==len)

{

printf("\n%d is not exist in the array",bf);

return;

}

for(;i<len;i++)

{

temp=arr[i];

arr[i]=n;

n=temp;

}

arr[i]=n;

len=len+1;

}

// Delete an element from the beginning

void del\_beg()

{

int i;

if(len==0)

{

printf("\nArray is empty.");

return;

}

printf("\n%d is Deleted.",arr[0]);

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

arr[i]=arr[i+1];

len = len - 1;

}

// Delete an element from the end

void del\_end()

{

if(len==0)

{

printf("\nArray is empty.");

return;

}

printf("\n%d is Deleted.",arr[len-1]);

len = len - 1;

}

// Delete any element

void del\_any()

{

int i,n;

if(len==0)

{

printf("\nArray is empty.");

return;

}

printf("\nEnter the element to delete = ");

scanf("%d",&n);

for(i=0;i<len && arr[i]!=n;i++);

if(i==len)

{

printf("\n%d is not found.",n);

return;

}

printf("\n%d is deleted.",arr[i]);

for(;i<len-1;i++)

arr[i]=arr[i+1];

len = len - 1;

}

// Delete an element after an element

void del\_after()

{

int i,n;

if(len==0)

{

printf("\nArray is empty.");

return;

}

printf("\nEnter the element whose after element have to delete = ");

scanf("%d",&n);

for(i=0;i<len && arr[i]!=n;i++);

if(i==len)

{

printf("\n%d is not found.",n);

return;

}

else

if(i==len-1)

{

printf("\n%d is the last element so farther deletion not possible.",n);

return;

}

printf("\n%d is deleted.",arr[i+1]);

for(i=i+1;i<len-1;i++)

arr[i]=arr[i+1];

len = len - 1;

}

// Delete an element before an element

void del\_before()

{

int i,n;

if(len==0)

{

printf("\nArray is empty.");

return;

}

printf("\nEnter the element whose before element have to delete = ");

scanf("%d",&n);

for(i=0;i<len && arr[i]!=n;i++);

if(i==len)

{

printf("\n%d is not found.",n);

return;

}

else

if(i==0)

{

printf("\n%d is the first element so farther deletion not possible.",n);

return;

}

printf("\n%d is deleted.",arr[i-1]);

for(i=i-1;i<len-1;i++)

arr[i]=arr[i+1];

len = len - 1;

}

// Display the array

void display()

{

int i;

printf("\nThe Elements are :-\n");

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

printf("A[%d] = %d\n",i+1,arr[i]);

}

/\* MAIN PROGRAM \*/

int main()

{

int n;

while(1)

{

printf("\n\n(1) Create an array.\n(2) Insert an element at first.\n(3) Insert an element at end.");

printf("\n(4) Insert an element at any position.\n(5) Insert an element after an element.");

printf("\n(6) Insert an element at before an element.\n(7) Delete an element from the beginning.");

printf("\n(8) Delete an element from end.\n(9) Delete any element.");

printf("\n(10) Delete any element after an element.\n(11) Delete an element before an element.");

printf("\n(12) Display.\n(13) Exit.");

printf("\n\nEnter Your choice = ");

scanf("%d",&n);

switch(n)

{

case 1:create\_array();

break;

case 2:insert\_beg();

break;

case 3:insert\_end();

break;

case 4:insert\_any();

break;

case 5:insert\_after();

break;

case 6:insert\_before();

break;

case 7:del\_beg();

break;

case 8:del\_end();

break;

case 9:del\_any();

break;

case 10:del\_after();

break;

case 11:del\_before();

break;

case 12:display();

break;

case 13:exit(0);

default:printf("\nWrong choice.........\n");

break;

}

}

return 0;

}

**Output**

susovan@susovan:~/Desktop/C Programs/Assignment$ gcc array.c

susovan@susovan:~/Desktop/C Programs/Assignment$ ./a.out

(1) Create an array.

(2) Insert an element at first.

(3) Insert an element at end.

(4) Insert an element at any position.

(5) Insert an element after an element.

(6) Insert an element at before an element.

(7) Delete an element from the beginning.

(8) Delete an element from end.

(9) Delete any element.

(10) Delete any element after an element.

(11) Delete an element before an element.

(12) Display.

(13) Exit.

Enter Your choice = 1

Enter the Number Of Elements (less then 50)= 10

Enter the Elements :-

A[1] = 3

A[2] = 4

A[3] = 2

A[4] = 9

A[5] = 1

A[6] = 6

A[7] = 0

A[8] = 3

A[9] = 6

A[10] = 34

(1) Create an array.

(2) Insert an element at first.

(3) Insert an element at end.

(4) Insert an element at any position.

(5) Insert an element after an element.

(6) Insert an element at before an element.

(7) Delete an element from the beginning.

(8) Delete an element from end.

(9) Delete any element.

(10) Delete any element after an element.

(11) Delete an element before an element.

(12) Display.

(13) Exit.

Enter Your choice = 12

The Elements are :-

A[1] = 3

A[2] = 4

A[3] = 2

A[4] = 9

A[5] = 1

A[6] = 6

A[7] = 0

A[8] = 3

A[9] = 6

A[10] = 34

(1) Create an array.

(2) Insert an element at first.

(3) Insert an element at end.

(4) Insert an element at any position.

(5) Insert an element after an element.

(6) Insert an element at before an element.

(7) Delete an element from the beginning.

(8) Delete an element from end.

(9) Delete any element.

(10) Delete any element after an element.

(11) Delete an element before an element.

(12) Display.

(13) Exit.

Enter Your choice = 2

Enter the element = 30

(1) Create an array.

(2) Insert an element at first.

(3) Insert an element at end.

(4) Insert an element at any position.

(5) Insert an element after an element.

(6) Insert an element at before an element.

(7) Delete an element from the beginning.

(8) Delete an element from end.

(9) Delete any element.

(10) Delete any element after an element.

(11) Delete an element before an element.

(12) Display.

(13) Exit.

Enter Your choice = 12

The Elements are :-

A[1] = 30

A[2] = 3

A[3] = 4

A[4] = 2

A[5] = 9

A[6] = 1

A[7] = 6

A[8] = 0

A[9] = 3

A[10] = 6

A[11] = 34

(1) Create an array.

(2) Insert an element at first.

(3) Insert an element at end.

(4) Insert an element at any position.

(5) Insert an element after an element.

(6) Insert an element at before an element.

(7) Delete an element from the beginning.

(8) Delete an element from end.

(9) Delete any element.

(10) Delete any element after an element.

(11) Delete an element before an element.

(12) Display.

(13) Exit.

Enter Your choice = 5

Enter the element to insert = 10

Enter the element after you want to insert 10 = 0

(1) Create an array.

(2) Insert an element at first.

(3) Insert an element at end.

(4) Insert an element at any position.

(5) Insert an element after an element.

(6) Insert an element at before an element.

(7) Delete an element from the beginning.

(8) Delete an element from end.

(9) Delete any element.

(10) Delete any element after an element.

(11) Delete an element before an element.

(12) Display.

(13) Exit.

Enter Your choice = 12

The Elements are :-

A[1] = 30

A[2] = 3

A[3] = 4

A[4] = 2

A[5] = 9

A[6] = 1

A[7] = 6

A[8] = 0

A[9] = 10

A[10] = 3

A[11] = 6

A[12] = 34

(1) Create an array.

(2) Insert an element at first.

(3) Insert an element at end.

(4) Insert an element at any position.

(5) Insert an element after an element.

(6) Insert an element at before an element.

(7) Delete an element from the beginning.

(8) Delete an element from end.

(9) Delete any element.

(10) Delete any element after an element.

(11) Delete an element before an element.

(12) Display.

(13) Exit.

Enter Your choice = 8

34 is Deleted.

(1) Create an array.

(2) Insert an element at first.

(3) Insert an element at end.

(4) Insert an element at any position.

(5) Insert an element after an element.

(6) Insert an element at before an element.

(7) Delete an element from the beginning.

(8) Delete an element from end.

(9) Delete any element.

(10) Delete any element after an element.

(11) Delete an element before an element.

(12) Display.

(13) Exit.

Enter Your choice = 7

30 is Deleted.

(1) Create an array.

(2) Insert an element at first.

(3) Insert an element at end.

(4) Insert an element at any position.

(5) Insert an element after an element.

(6) Insert an element at before an element.

(7) Delete an element from the beginning.

(8) Delete an element from end.

(9) Delete any element.

(10) Delete any element after an element.

(11) Delete an element before an element.

(12) Display.

(13) Exit.

Enter Your choice = 12

The Elements are :-

A[1] = 3

A[2] = 4

A[3] = 2

A[4] = 9

A[5] = 1

A[6] = 6

A[7] = 0

A[8] = 10

A[9] = 3

A[10] = 6

(1) Create an array.

(2) Insert an element at first.

(3) Insert an element at end.

(4) Insert an element at any position.

(5) Insert an element after an element.

(6) Insert an element at before an element.

(7) Delete an element from the beginning.

(8) Delete an element from end.

(9) Delete any element.

(10) Delete any element after an element.

(11) Delete an element before an element.

(12) Display.

(13) Exit.

Enter Your choice = 9

Enter the element to delete = 10

10 is deleted.

(1) Create an array.

(2) Insert an element at first.

(3) Insert an element at end.

(4) Insert an element at any position.

(5) Insert an element after an element.

(6) Insert an element at before an element.

(7) Delete an element from the beginning.

(8) Delete an element from end.

(9) Delete any element.

(10) Delete any element after an element.

(11) Delete an element before an element.

(12) Display.

(13) Exit.

Enter Your choice = 4

Enter the element = 4

Enter the Position =2

(1) Create an array.

(2) Insert an element at first.

(3) Insert an element at end.

(4) Insert an element at any position.

(5) Insert an element after an element.

(6) Insert an element at before an element.

(7) Delete an element from the beginning.

(8) Delete an element from end.

(9) Delete any element.

(10) Delete any element after an element.

(11) Delete an element before an element.

(12) Display.

(13) Exit.

Enter Your choice = 12

The Elements are :-

A[1] = 3

A[2] = 4

A[3] = 4

A[4] = 2

A[5] = 9

A[6] = 1

A[7] = 6

A[8] = 0

A[9] = 3

A[10] = 6

(1) Create an array.

(2) Insert an element at first.

(3) Insert an element at end.

(4) Insert an element at any position.

(5) Insert an element after an element.

(6) Insert an element at before an element.

(7) Delete an element from the beginning.

(8) Delete an element from end.

(9) Delete any element.

(10) Delete any element after an element.

(11) Delete an element before an element.

(12) Display.

(13) Exit.

Enter Your choice = 11

Enter the element whose before element have to delete = 3

3 is the first element so farther deletion not possible.

(1) Create an array.

(2) Insert an element at first.

(3) Insert an element at end.

(4) Insert an element at any position.

(5) Insert an element after an element.

(6) Insert an element at before an element.

(7) Delete an element from the beginning.

(8) Delete an element from end.

(9) Delete any element.

(10) Delete any element after an element.

(11) Delete an element before an element.

(12) Display.

(13) Exit.

Enter Your choice = 11

Enter the element whose before element have to delete = 0

6 is deleted.

(1) Create an array.

(2) Insert an element at first.

(3) Insert an element at end.

(4) Insert an element at any position.

(5) Insert an element after an element.

(6) Insert an element at before an element.

(7) Delete an element from the beginning.

(8) Delete an element from end.

(9) Delete any element.

(10) Delete any element after an element.

(11) Delete an element before an element.

(12) Display.

(13) Exit.

Enter Your choice = 12

The Elements are :-

A[1] = 3

A[2] = 4

A[3] = 4

A[4] = 2

A[5] = 9

A[6] = 1

A[7] = 0

A[8] = 3

A[9] = 6

(1) Create an array.

(2) Insert an element at first.

(3) Insert an element at end.

(4) Insert an element at any position.

(5) Insert an element after an element.

(6) Insert an element at before an element.

(7) Delete an element from the beginning.

(8) Delete an element from end.

(9) Delete any element.

(10) Delete any element after an element.

(11) Delete an element before an element.

(12) Display.

(13) Exit.

Enter Your choice = 13

susovan@susovan:~/Desktop/C Programs/Assignment$

**ASSIGNMENT - 2**

**Problem Statement**

Write a menu driven program to implement a singly Linked list with different type of operations: Insertion, Deletion, Traverse, Searching.

Insertion and Deletion have different way of execution enlisted following:

1. At the beginning
2. At the end
3. At any given position
4. After an element
5. Before an element

**Solution**

**Theory**

Here we use linked list as the Data Structure. The linked list is pointed by the pointer **‘h’**, every node of the linked list has two part one is data part called x holds the data item and another is address part called next that hold the next node address or null.

**Algorithm**

create\_list()

Begin

h🡨null

q🡨h

ch🡨 ‘y’

while ch = ‘Y’ or ch = ‘y’ then

do

print “Enter the element”

input x

create a node with data item x and save its address into the p

if h= null

then

h🡨p

else

next[q]🡨p

end if

q🡨p

print “Do you want to continue ”

input ch

end while

End

insert\_any()

Begin

if h=null

then

print “List not exist”

return

end if

c🡨1

Print “Enter the Number”

input x

print “position”

input pos

create a node with data item x and save its address into the p

if pos= 1

then

next[p]🡨h

h🡨p

return

end if

q🡨h

while next[q] not equal to null and c<pso-1 then

do

q🡨next[q]

c🡨c+1

end while

next[p]🡨next[q]

next[q]🡨p

End

del\_any()

Begin

if h=null

then

print “List not exist”

return

end if

print “Enter the element to delete”

input x

p🡨q🡨h

while x[q] not equal to x and next[q] not equal to null then

do

p🡨q

q🡨next[q]

end while

if x[q] not equal to x

then

print “Element not found”

return

end if

print x is deleted

if x[h] = x

then

h🡨next[h]

else

next[p]🡨next[q]

end if

free the node that location is blocked by q

End

Display()

Begin

p🡨h

if p=null

then

print “List not exist”

return

end if

print “elements are”

while p not equals to null then

do

print x[p]

p🡨next[p]

end while

End

function “Main()”

Begin

choice one of the operation to perform on the ‘arr’

1. create\_list()
2. insert\_any()
3. del\_any()
4. display()

End

**Source Code**

#include<stdio.h>

#include<malloc.h>

#include<stdlib.h>

// Structure declaration

typedef struct node

{

int x;

struct node \*next;

}node;

node \*h; // Pointer to the head of List

// Creating the Node of the List

node \*getnode(int x)

{

node \*p;

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

p->x=x;

p->next=NULL;

return(p);

}

// Create a list

void create\_list()

{

node \*p,\*q;

int x;

char ch;

h=NULL;

q=h;

do

{

printf("\nEnter a Element =");

scanf("%d",&x);

p=getnode(x);

if(h==NULL)

h=p;

else

q->next=p;

q=p;

printf("\nDo you want to continue(Y/N)=");

fflush(stdin);

scanf("%s",&ch);

}while(ch=='y'||ch=='Y');

}

// Insert an element at beg

void insert\_beg()

{

node \*p;

int x;

if(h==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\nEnter a Element =");

scanf("%d",&x);

p=getnode(x);

p->next=h;

h=p;

}

// Insert an element at end

void insert\_end()

{

node \*p,\*q;

int x;

if(h==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\nEnter the Element =");

scanf("%d",&x);

p=getnode(x);

q=h;

while(q->next!=NULL)

q=q->next;

q->next=p;

}

// Insert an element at any Position

void insert\_mid()

{

node \*p,\*q;

int x,c=1,pos;

if(h==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\nEnter the Element =");

scanf("%d",&x);

printf("\nEnter the Position =");

scanf("%d",&pos);

p=getnode(x);

if(pos==1)

{

p->next=h;

h=p;

return;

}

q=h;

while(q->next!=NULL&&c<pos-1)

{

q=q->next;

c++;

}

p->next=q->next;

q->next=p;

}

// Insert an element before an element

void insert\_pre()

{

node \*p,\*q;

int x,n;

if(h==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\nEnter the Element =");

scanf("%d",&x);

printf("\nEnter the Element whose before you want to insert %d = ",x);

scanf("%d",&n);

p=h;

while(p->next!=NULL && p->x!=n)

{

q=p;

p=p->next;

}

if(p==h)

{

p=getnode(x);

p->next=h;

h=p;

return;

}

q->next=getnode(x);

q->next->next=p;

}

// Insert an element after an element

void insert\_next()

{

node \*p,\*q;

int x,n;

if(h==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\nEnter the Element =");

scanf("%d",&x);

printf("\nEnter the Element whose after you want to insert %d = ",x);

scanf("%d",&n);

p=h;

while(p->next!=NULL && p->x!=n)

p=p->next;

q=p->next;

p->next=getnode(x);

p->next->next=q;

}

// Delete an element from the beginning

void del\_beg()

{

node \*p;

if(h==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\n%d is deleted.",h->x);

p=h;

h=h->next;

free(p);

}

// Delete an element from the end

void del\_end()

{

node \*p,\*q;

if(h==NULL)

{

printf("\nList Not Exist.");

return;

}

p=q=h;

while(q->next!=NULL)

{

p=q;

q=q->next;

}

printf("\n%d is deleted.",q->x);

if(p==q)

h=NULL;

else

p->next=NULL;

free(q);

}

// Delete any element

void del\_any()

{

node\*p,\*q;

int x;

if(h==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\nEnter the Element to Delete =");

scanf("%d",&x);

p=q=h;

while(q->x!=x&&q->next!=NULL)

{

p=q;

q=q->next;

}

if(q->x!=x)

{

printf("\nElement not Found.");

return;

}

printf("\n%d is deleted.",x);

if(h->x==x)

h=h->next;

else

p->next=q->next;

free(q);

}

// Delete an element before an element

void del\_pre()

{

node \*p,\*q,\*r;

int x;

printf("\nEnter the Element whose previous Element have to Delete =");

scanf("%d",&x);

if(h->x==x)

{

printf("\nDeletion Not possible.");

return;

}

p=q=h;

r=h->next;

while(r!=NULL&&r->x!=x)

{

p=q;

q=r;

r=r->next;

}

if(r==NULL)

{

printf("\nElement Not Found.");

return;

}

if(p==q)

h=h->next;

else

p->next=r;

free(q);

}

// Delete an element after an element

void del\_next()

{

node \*p,\*q;

int x;

if(h==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\nEnter the Element whose Next Element have to Delete =");

scanf("%d",&x);

p=h;

q=h->next;

while(p->x!=x&&q!=NULL)

{

p=q;

q=q->next;

}

if(q==NULL)

{

printf("\nDeletion Not Possible.");

return;

}

p->next=q->next;

free(q);

}

void display()

{

node \*p;

p=h;

if(p==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\nThe Elements :-\n");

while(p!=NULL)

{

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

p=p->next;

}

}

int main()

{

int n;

while(1)

{

printf("\n\n1 for Create a List.");

printf("\n2 for Insert an Element in the List at Beginning.");

printf("\n3 for Insert an Element in the List at Any Position.");

printf("\n4 for Insert an Element in the List at End.");

printf("\n5 for Insert an Element Before the given Element.");

printf("\n6 for Insert an Element After the given Element.");

printf("\n7 for Delete an Element from the Beginning of the List.");

printf("\n8 for Delete an Element from the End of the List.");

printf("\n9 for Delete any Element from the List.");

printf("\n10 for Delete the Previous Element of the given Element from the List.");

printf("\n11 for Delete the Next Element of the given Element from the List.");

printf("\n12 for Display the List.");

printf("\n0 for EXIT.");

printf("\nEnter your Choice =");

scanf("%d",&n);

switch(n)

{

case 1:create\_list();

break;

case 2:insert\_beg();

break;

case 3:insert\_mid();

break;

case 4:insert\_end();

break;

case 5:insert\_pre();

break;

case 6:insert\_next();

break;

case 7:del\_beg();

break;

case 8:del\_end();

break;

case 9:del\_any();

break;

case 10:del\_pre();

break;

case 11:del\_next();

break;

case 12:display();

break;

case 0:exit(0);

default:printf("\nWrong Choice........");

}

}

}

**Output**

susovan@susovan:~/Desktop/C Programs/Assignment$ gcc linked\_list\_S.c

susovan@susovan:~/Desktop/C Programs/Assignment$ ./a.out

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =1

Enter a Element =6

Do you want to continue(Y/N)=y

Enter a Element =5

Do you want to continue(Y/N)=y

Enter a Element =9

Do you want to continue(Y/N)=y

Enter a Element =12

Do you want to continue(Y/N)=n

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =12

The Elements :-

6 5 9 12

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =2

Enter a Element =4

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =12

The Elements :-

4 6 5 9 12

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =4

Enter the Element =56

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =12

The Elements :-

4 6 5 9 12 56

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =5

Enter the Element =38

Enter the Element whose before you want to insert 38 = 1

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =12

The Elements :-

4 6 5 9 12 38 56

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =7

4 is deleted.

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =8

56 is deleted.

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =12

The Elements :-

6 5 9 12 38

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =9

Enter the Element to Delete =9

9 is deleted.

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =12

The Elements :-

6 5 12 38

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =11

Enter the Element whose Next Element have to Delete =12

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =12

The Elements :-

6 5 12

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =0

susovan@susovan:~/Desktop/C Programs/Assignment$

**ASSIGNMENT - 3**

**Problem Statement**

Write a menu driven program to implement a doubly Linked list with different type of operations: Insertion, Deletion, Traverse, Searching.

Insertion and Deletion have different way of execution enlisted following:

1. At the beginning
2. At the end
3. At any given position
4. After an element
5. Before an element

**Solution**

**Theory**

Here we use double linked list as the Data Structure. The linked list is pointed by the pointer **‘h’**, every node of the list has three part one is data part called x holds the data item and others are address part called next that hold the next node address and pre that hold the previous node address.

**Algorithm**

create\_list()

Begin

q🡨 pre[h]🡨next[h]🡨null

ch🡨 ‘y’

while ch=’y’ or ch = ‘Y’ then

do

print “enter the element “

input x

create a node with data item x and save its address into p

if pre[h] = null

then

pre[h] 🡨 p

else

next[q]🡨p

pre[p]🡨q

end if

q🡨p

print “Do you want to continue”

input ch

end while

next[h] 🡨 p

End

insert\_any()

Begin

if pre[h] = null

then

print “list not exist”

return

end if

print “Enter the element”

input x

print “Enter the position”

input pos

c🡨1

create a node with data item x and save its address into p

if pos=1

then

next[p] 🡨 pre[h]

pre[pre[h]] 🡨 p

pre[h] 🡨 p

return

end if

q🡨pre[h]

while next[q] not equal to null and c<pos-1 then

do

q🡨next[q]

c🡨c+1

end while

if next[q] = null

then

pre[h] 🡨 next[h]

next[next[h]] 🡨 p

next[p] 🡨 p

return

end if

next[p]🡨next[q]

pre[next[q]] 🡨 p

pre[p] 🡨 q

next[q] 🡨 p

End

del\_any()

begin

if pre[h] = null

then

print “list not exist”

return

end if

print “Enter the element to delete”

input x

p🡨q🡨pre[h]

while x[q] not equal to x and next[q] not equal to null then

do

q🡨next[q]

end while

if x[q] not equal to x

then

print “Element not found”

return

end if

print x is deleted

if pre[h]=next[h]

then

next[h]🡨pre[h]🡨null

else

if x[pre[h]]=x

then

pre[h]🡨next[pre[h]]

else

if next[q] = null

then

next[h]🡨pre[next[h]]

next[next[h]]🡨null

else

pre[next[q]] 🡨pre[q]

next[pre[q]]🡨next[q]

end if

end if

end if

free the location q

End

display()

Begin

if pre[h] = null

then

print “list not exist”

return

end if

print “1 for forward display”

print “2 for backward display”

print “Enter your choice”

input ch

if ch = 1

then

p🡨pre[h]

while p not equal to null then

do

print x[p]

p🡨next[p]

end while

else

p🡨next[h]

while p not equal to null then

do

print x[p]

p🡨pre[p]

end while

end if

End

function “Main()”

Begin

choice one of the operation to perform on the ‘arr’

1. create\_list()
2. insert\_any()
3. del\_any()
4. display()

End

**Source Code**

#include<stdio.h>

#include<malloc.h>

#include<stdlib.h>

// Structure declaration

typedef struct node

{

int x;

struct node \*next,\*pre;

}node;

node \*h; // Pointer to the head of List

// Creating the Node of the List

node \*getnode(int x)

{

node \*p;

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

p->x=x;

p->next=NULL;

p->pre=NULL;

return(p);

}

// Create a list

void create\_list()

{

node \*p,\*q;

int x;

char ch;

q = h->pre = h->next = NULL;

do

{

printf("\nEnter a Element =");

scanf("%d",&x);

p=getnode(x);

if(h->pre==NULL)

h->pre=p;

else

{

q->next=p;

p->pre=q;

}

q=p;

printf("\nDo you want to continue(Y/N)=");

fflush(stdin);

scanf("%s",&ch);

}while(ch=='y'||ch=='Y');

h->next=p;

}

// Insert an element at beg

void insert\_beg()

{

node \*p;

int x;

if(h->pre==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\nEnter a Element =");

scanf("%d",&x);

p=getnode(x);

p->next=h->pre;

h->pre->pre=p;

h->pre=p;

}

// Insert an element at end

void insert\_end()

{

node \*p;

int x;

if(h->pre==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\nEnter the Element =");

scanf("%d",&x);

p=getnode(x);

p->pre=h->next;

h->next->next=p;

h->next=p;

}

// Insert an element at any Position

void insert\_mid()

{

node \*p,\*q;

int x,c=1,pos;

if(h->pre==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\nEnter the Element =");

scanf("%d",&x);

printf("\nEnter the Position =");

scanf("%d",&pos);

p=getnode(x);

if(pos==1)

{

p->next=h->pre;

h->pre->pre=p;

h->pre=p;

return;

}

q=h->pre;

while(q->next!=NULL&&c<pos-1)

{

q=q->next;

c++;

}

if(q->next == NULL)

{

p->pre=h->next;

h->next->next=p;

h->next=p;

return;

}

p->next=q->next;

q->next->pre=p;

p->pre=q;

q->next=p;

}

// Insert an element before an element

void insert\_pre()

{

node \*p,\*q;

int x,n;

if(h->pre==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\nEnter the Element =");

scanf("%d",&x);

printf("\nEnter the Element whose before you want to insert %d = ",x);

scanf("%d",&n);

q=h->pre;

while(q->next!=NULL && q->x!=n)

q=q->next;

if(q->x != n)

{

printf("\n%d is not in the List.\n",n);

return;

}

p=getnode(x);

if(q==h->pre)

{

p->next=h->pre;

h->pre->pre=p;

h->pre=p;

return;

}

p->next = q;

p->pre = q->pre;

q->pre->next = p;

q->pre = p;

}

// Insert an element after an element

void insert\_next()

{

node \*p,\*q;

int x,n;

if(h->pre==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\nEnter the Element =");

scanf("%d",&x);

printf("\nEnter the Element whose after you want to insert %d = ",x);

scanf("%d",&n);

p=h->pre;

while(p->next!=NULL && p->x!=n)

p=p->next;

if(p->x != n)

{

printf("\n%d is not in the List.\n",n);

return;

}

if(p->next == NULL)

{

p=getnode(x);

p->pre=h->next;

h->next->next=p;

h->next=p;

return;

}

q=getnode(x);

q->next=p->next;

p->next->pre=q;

p->next=q;

q->pre=p;

}

// Delete an element from the beginning

void del\_beg()

{

node \*p;

if(h->pre==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\n%d is deleted.",h->pre->x);

p=h->pre;

h->pre=h->pre->next;

free(p);

}

// Delete an element from the end

void del\_end()

{

node \*p;

if(h->pre==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\n%d is deleted.",h->next->x);

p=h->next;

if(h->next == h->pre)

h->next = h->pre = NULL;

else

{

h->next = h->next->pre;

h->next->next = NULL;

}

free(p);

}

// Delete any element

void del\_any()

{

node\*p,\*q;

int x;

if(h->pre==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\nEnter the Element to Delete =");

scanf("%d",&x);

p=q=h->pre;

while(q->x!=x&&q->next!=NULL)

q=q->next;

if(q->x!=x)

{

printf("\nElement not Found.");

return;

}

printf("\n%d is deleted.",x);

if(h->next == h->pre)

h->next = h->pre = NULL;

else

{

if(h->pre->x==x)

h->pre=h->pre->next;

else

{

if(q->next == NULL)

{

h->next = h->next->pre;

h->next->next = NULL;

}

else

{

q->next->pre = q->pre;

q->pre->next = q->next;

}

}

}

free(q);

}

// Delete an element before an element

void del\_pre()

{

node \*r;

int x;

printf("\nEnter the Element whose previous Element have to Delete =");

scanf("%d",&x);

if(h->pre->x==x)

{

printf("\nDeletion Not possible.");

return;

}

r=h->pre;

while(r!=NULL&&r->x!=x)

r=r->next;

if(r==NULL)

{

printf("\nElement Not Found.");

return;

}

r=r->pre;

if(r->pre == NULL)

h->pre = h->pre->next;

else

{

r->next->pre = r->pre;

r->pre->next = r->next;

}

free(r);

}

// Delete an element after an element

void del\_next()

{

node \*p,\*q;

int x;

if(h->pre==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\nEnter the Element whose Next Element have to Delete =");

scanf("%d",&x);

q=h->pre;

while(q->x!=x&&q->next!=NULL)

q=q->next;

if(q->next==NULL)

{

printf("\nDeletion Not Possible.");

return;

}

if(q->next->next==NULL)

{

q->next=NULL;

h->next=h->next->pre;

}

else

{

q=q->next;

q->pre->next=NULL;

}

free(q);

}

// Display the list

void display()

{

node \*p;

int n;

if(h->pre==NULL)

{

printf("\nList Not Exist.");

return;

}

printf("\nDisplay Order\n1 for Forward Display.\n2 for Backward Display.\nEnter Your Choice = ");

scanf("%d",&n);

printf("\nThe Elements :-\n");

switch(n)

{

case 1: p=h->pre;

while(p!=NULL)

{

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

p=p->next;

}

break;

case 2: p=h->next;

while(p!=NULL)

{

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

p=p->pre;

}

break;

default:printf("\nWrong Choice........");

}

}

int main()

{

int n;

h=(node\*)malloc(sizeof(node));

while(1)

{

printf("\n\n1 for Create a List.");

printf("\n2 for Insert an Element in the List at Beginning.");

printf("\n3 for Insert an Element in the List at Any Position.");

printf("\n4 for Insert an Element in the List at End.");

printf("\n5 for Insert an Element Before the given Element.");

printf("\n6 for Insert an Element After the given Element.");

printf("\n7 for Delete an Element from the Beginning of the List.");

printf("\n8 for Delete an Element from the End of the List.");

printf("\n9 for Delete any Element from the List.");

printf("\n10 for Delete the Previous Element of the given Element from the List.");

printf("\n11 for Delete the Next Element of the given Element from the List.");

printf("\n12 for Display the List.");

printf("\n0 for EXIT.");

printf("\nEnter your Choice =");

scanf("%d",&n);

switch(n)

{

case 1:create\_list();

break;

case 2:insert\_beg();

break;

case 3:insert\_mid();

break;

case 4:insert\_end();

break;

case 5:insert\_pre();

break;

case 6:insert\_next();

break;

case 7:del\_beg();

break;

case 8:del\_end();

break;

case 9:del\_any();

break;

case 10:del\_pre();

break;

case 11:del\_next();

break;

case 12:display();

break;

case 0:exit(0);

default:printf("\nWrong Choice........");

}

}

}

**Output**

susovan@susovan:~/Desktop/C Programs/Assignment$ gcc linked\_list\_D.c

susovan@susovan:~/Desktop/C Programs/Assignment$ ./a.out

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =1

Enter a Element =7

Do you want to continue(Y/N)=y

Enter a Element =4

Do you want to continue(Y/N)=y

Enter a Element =2

Do you want to continue(Y/N)=n

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =12

Display Order

1 for Forward Display.

2 for Backward Display.

Enter Your Choice = 1

The Elements :-

7 4 2

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =12

Display Order

1 for Forward Display.

2 for Backward Display.

Enter Your Choice = 2

The Elements :-

2 4 7

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =2

Enter a Element =23

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =12

Display Order

1 for Forward Display.

2 for Backward Display.

Enter Your Choice = 1

The Elements :-

23 7 4 2

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =4

Enter the Element =20

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =12

Display Order

1 for Forward Display.

2 for Backward Display.

Enter Your Choice = 1

The Elements :-

23 7 4 2 20

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =5

Enter the Element =45

Enter the Element whose before you want to insert 45 = 4

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =12

Display Order

1 for Forward Display.

2 for Backward Display.

Enter Your Choice = 1

The Elements :-

23 7 45 4 2 20

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =7

23 is deleted.

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =8

20 is deleted.

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =12

Display Order

1 for Forward Display.

2 for Backward Display.

Enter Your Choice = 1

The Elements :-

7 45 4 2

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =3

Enter the Element =46

Enter the Position =3

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =12

Display Order

1 for Forward Display.

2 for Backward Display.

Enter Your Choice = 1

The Elements :-

7 45 46 4 2

1 for Create a List.

2 for Insert an Element in the List at Beginning.

3 for Insert an Element in the List at Any Position.

4 for Insert an Element in the List at End.

5 for Insert an Element Before the given Element.

6 for Insert an Element After the given Element.

7 for Delete an Element from the Beginning of the List.

8 for Delete an Element from the End of the List.

9 for Delete any Element from the List.

10 for Delete the Previous Element of the given Element from the List.

11 for Delete the Next Element of the given Element from the List.

12 for Display the List.

0 for EXIT.

Enter your Choice =0

susovan@susovan:~/Desktop/C Programs/Assignment$

**ASSIGNMENT - 4**

**Problem Statement**

Write a menu driven program to implement quick sort and merge sort

**Solution**

**Theory**

Here we use array as data structure named as **‘a’**. Two sorting algorithms are here, we can sue one of them to sort the array a.

**Algorithm**

// Quick sort Algorithm start

partition(a,lb,ub)

Begin

pivot🡨a[lb]

start🡨lb

end🡨ub

while start < end then

do

while a[start] <= pivot then

do

start🡨start+1

end while

while a[end] > pivot then

do

end🡨end-1

end while

if start < end

then

temp🡨a[end]

a[end]🡨a[start]

a[start]🡨temp

end if

end while

temp🡨a[end]

a[end]🡨a[lb]

a[lb]🡨temp

return end

End

quick\_sort(a,lb,ub)

Begin

if lb<ub

then

part🡨 partition(a,lb,ub)

quick\_sort(a,lb,part-1)

quick\_sort(a,part+1,ub)

end if

End

// Merge sort algorithm start

merge(a,lb,mid,ub)

Begin

create a array of size (ub-lb+1) and named b

i🡨lb

j🡨mid+1

k🡨lb

while i<=mid and j<= ub then

do

if a[i]<=a[j]

then

b[k] 🡨a[i]

k🡨k+1

i🡨i+1

else

b[k] 🡨a[j]

k🡨k+1

j🡨j+1

end if

end while

while i<=mid then

do

b[k] 🡨a[i]

k🡨k+1

i🡨i+1

end while

while j<=ub then

do

b[k] 🡨a[j]

k🡨k+1

j🡨j+1

end while

k🡨lb

while k<=ub then

do

a[k] 🡨b[k]

k🡨k+1

end while

End

merge\_sort(a,lb,ub)

Begin

if lb<ub

then

mid 🡨 (lb+ub)/2

merge\_sort(a,lb,mid)

merge\_sort(a,mid+1,ub)

merge(a,lb,mid,ub)

end if

End

Function “Main()”

Begin

print “Enter the size of array”

input n

create the array of size of n and named ‘a’

print “Enter the Elements ”

input a

print “elements are”

print a

print “1 for quick sort 2 for merge sort”

print “enter Your choice”

input ch

if ch = 1

then

quick\_sort(a,0,n-1)

else

merge\_sort(a,0,n-1)

end if

print “Elements after Sort”

print a

End

**Source Code**

#include<stdio.h>

#include<stdlib.h>

// quick sort start

int partition(int \*a,int lb,int ub)

{

int start,end,pivot,temp;

pivot=a[lb];

start=lb;

end=ub;

while(start<end)

{

while(a[start]<=pivot)

start++;

while(a[end]>pivot)

end--;

if(start<end)

{

temp=a[end];

a[end]=a[start];

a[start]=temp;

}

}

temp=a[end];

a[end]=a[lb];

a[lb]=temp;

return end;

}

void quick\_sort(int \*a,int lb,int ub)

{

int part;

if(lb<ub)

{

part = partition(a,lb,ub);

quick\_sort(a,lb,part-1);

quick\_sort(a,part+1,ub);

}

}

// quick sort end

// merge sort start

void merge(int \*a,int lb,int mid,int ub)

{

int i,j,k,\*b;

b=(int\*)malloc(sizeof(int)\*(ub-lb+1));

i=lb;

j=mid+1;

k=lb;

while(i<=mid && j<=ub)

{

if(a[i]<=a[j])

b[k++]=a[i++];

else

b[k++]=a[j++];

}

while(i<=mid)

b[k++]=a[i++];

while(j<=ub)

b[k++]=a[j++];

for(k=lb;k<=ub;k++)

a[k]=b[k];

}

void merge\_sort(int \*a,int lb,int ub)

{

int mid;

if(lb<ub)

{

mid = (lb+ub)/2;

merge\_sort(a,lb,mid);

merge\_sort(a,mid+1,ub);

merge(a,lb,mid,ub);

}

}

// merge sort end

int main()

{

int \*a,n,i,ch;

printf("\nEnter the Number of the array elements = ");

scanf("%d",&n);

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

printf("\nEnter the Elements :- \n");

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

{

printf("Arr[%d] = ",i);

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

}

printf("\nThe Elements are :- \n");

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

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

printf("\n\n1 for quick sort.\n2 for merge sort.\nEnter Your choice = ");

scanf("%d",&ch);

if(ch == 1)

quick\_sort(a,0,n-1);

else

if(ch == 2)

merge\_sort(a,0,n-1);

else

{

printf("\nwrong choice....");

exit(0);

}

printf("\nThe Elements are after sorting :- \n");

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

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

}

**Output**

susovan@susovan:~/Desktop/C Programs/Assignment$ gcc sort\_qm.c

susovan@susovan:~/Desktop/C Programs/Assignment$ ./a.out

Enter the Number of the array elements = 5

Enter the Elements :-

Arr[0] = 5

Arr[1] = 4

Arr[2] = 3

Arr[3] = 2

Arr[4] = 1

The Elements are :-

5 4 3 2 1

1 for quick sort.

2 for merge sort.

Enter Your choice = 1

The Elements are after sorting :-

1 2 3 4 5

susovan@susovan:~/Desktop/C Programs/Assignment$ ./a.out

Enter the Number of the array elements = 6

Enter the Elements :-

Arr[0] = 5

Arr[1] = 3

Arr[2] = 6

Arr[3] = 3

Arr[4] = 7

Arr[5] = 2

The Elements are :-

5 3 6 3 7 2

1 for quick sort.

2 for merge sort.

Enter Your choice = 2

The Elements are after sorting :-

2 3 3 5 6 7

susovan@susovan:~/Desktop/C Programs/Assignment$

**ASSIGNMENT – 5**

**Problem Statement**

Implementing Shell sort with suitable elements.

**Solution**

**Theory**

Here we use array as the data structure and named it **‘a’**. We sort the elements of the array a by the Shell sort.

**Algorithm**

shell\_sort(a,gap,n)

Begin

if gap = 0

then

return

end if

i🡨0

j🡨i+gap

while j<n then

do

t1🡨i

t2🡨j

while t1>=0 then

do

if a[t1] > a[t2]

then

swap the value of a[t1] with a[t2]

end if

t2🡨t1

t1🡨t1-gap

end while

j🡨j+1

i🡨i+1

end while

shell\_sort(a,gap/2,n)

End

function “main()”

Begin

print “Enter the Number of element”

Input n

create a array of size n and named a

print “Enter the elements”

input a

print “Elements before sort”

print a

shell\_sort(a,n/2,n)

print “elements after sort”

print a

End

**Source Code**

#include<stdio.h>

#include<stdlib.h>

void swap(int \*a,int \*b)

{

int temp;

temp = \*a;

\*a = \*b;

\*b = temp;

}

void shell\_sort(int \*a,int gap,int n)

{

int i,j,t1,t2;

if(gap == 0)

return;

for(i=0,j=i+gap;j<n;j++,i++)

{

t1=i;

t2=j;

while(t1>=0)

{

if(a[t1]>a[t2])

swap(&a[t1],&a[t2]);

t2=t1;

t1=t1-gap;

}

}

shell\_sort(a,gap/2,n);

}

int main()

{

int \*a,n,i;

printf("\nEnter the Number of the elements = ");

scanf("%d",&n);

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

printf("\nEnter the Elements :-\n");

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

{

printf("arr[%d] = ",i);

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

}

printf("\nThe Elements Before Sort :- \n");

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

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

shell\_sort(a,n/2,n);

printf("\nThe Elements After Sort :- \n");

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

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

}

**Output**

susovan@susovan:~/Desktop/C Programs/Assignment$ gcc sort\_shell.c

susovan@susovan:~/Desktop/C Programs/Assignment$ ./a.out

Enter the Number of the elements = 10

Enter the Elements :-

arr[0] = 5

arr[1] = 78

arr[2] = 4

arr[3] = 87

arr[4] = 38

arr[5] = 23

arr[6] = 865

arr[7] = 4

arr[8] = 87

arr[9] = 43

The Elements Before Sort :-

5 78 4 87 38 23 865 4 87 43

The Elements After Sort :-

4 4 5 23 38 43 78 87 87 865

susovan@susovan:~/Desktop/C Programs/Assignment$

**ASSIGNMENT - 6**

**Problem Statement**

Implement sieve sort with suitable elements.

**Solution**

**Theory**

Here we use array data structure and named as **‘a’**. We implemented Sieve sort on the array a.

**ALGORITHM**

int sieve(int \*\*b, int a[],int y)

begin

STEP 1: [INITIALISE THE VARIABLES.]

int i, j, k=1, subtree=1, f, flag=1;

b[0][0]=a[0];

STEP 2: Repeat while( k < n )

2.1:[INITIALISE THE VARIABLES]

i=0, f=0;

2.2: Repeat while( i < subtree && f!=1 )

2.2.1: j=0;

2.2.2: if( b[i][0] > a[k] ) then

do

2.2.2.1: Repeat while( b[i][j]!=' ' )

2.2.2.1.1: j = j + 1

END OF STEP 2.2.2.1.

2.2.2.2: b[i][j] = a[k]

2.2.2.3: f=1

END OF STEP 2.2.2.

2.2.3: i = i + 1;

END OF STEP 2.2.

2.3: if( f==0 ) then

do

2.3.1: subtree = subtree + 1

2.3.2: b[i][0] = a[k]

END OF STEP 2.3.

2.4: k = k + 1;

END OF STEP 2.

STEP 3: f = 0;

STEP 4: Repeat for ( i=0 to i<subtree)

4.1: k=0;

4.2: Repeat while( b[i][k]!=' ' )

4.2.1: k = k + 1;

END OF STEP 4.2.

4.3: Repeat for ( j=k-1 to j>=0)

4.3.1: a[f++] = b[i][j];

4.3.2: j = j – 1

END OF STEP 4.3.

4.4: print ("SUBLIST- : ", i+1)

4.5: Repeat for( j=0 to j<k)

4.5.1: if( y==0 ) then print the value of ( b[i][j] )

4.5.2: else print the value of ( b[i][k-j-1] )

4.5.3: j = j + 1

END OF STEP 4.5

END OF STEP 4.

STEP 5: Repeat for( i=0 to i<n )

5.1: print the value of( a[i] )

5.2: i = i + 1

END OF STEP 5.

STEP 6: Repeat for ( i=0 to i<n-1 )

6.1: if ( a[i] > a[i+1] ) then

6.1.1: flag = 0;

6.1.2: break;

END OF STEP 6.1.

END OF STEP 6.

STEP 7: return ( flag )

end

**Source Code**

#include<stdio.h>

#include<malloc.h>

int n;

int sieve(int \*\*b, int a[],int y)

{

int i,j,k=1,subtree=1,f,flag=1;

b[0][0]=a[0];

while(k<n)

{

i=0;

f=0;

while(i<subtree && f!=1)

{

j=0;

if(b[i][0]>a[k])

{

while(b[i][j]!=' ')

j++;

b[i][j]=a[k];

f=1;

}

i++;

}

if(f==0)

{

subtree++;

b[i][0]=a[k];

}

k++;

}

f=0;

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

{

k=0;

while(b[i][k]!=' ')

k++;

for(j=k-1;j>=0;j--)

a[f++]=b[i][j];

printf("SUBLIST- %d : ",i+1);

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

{

if(y==0)

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

else

printf("%d ",b[i][k-j-1]);

}

printf("\n");

}

printf("\n");

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

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

printf("\n");

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

{

if(a[i]>a[i+1])

{

flag=0;

break;

}

}

return(flag);

}

main()

{

int \*a,\*\*b,i,j,k,t,x=0;

printf("Enter the number of elements : ");

scanf("%d",&n);

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

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

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

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

printf("Enter the elements :\n");

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

{

printf("a[%d] : ",i+1);

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

}

i=0;

while(x!=1)

{

printf("\nPASS- %d\n",i+1);

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

{

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

b[j][k]=' ';

}

if(i%2==0)

x=sieve(b,a,0);

else

x=sieve(b,a,1);

i++;

}

printf("\nSORTED ELEMENTS ARE :\n");

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

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

}

**OUTPUT**

Enter the number of elements : 6

Enter the elements :

a[1] : 78

a[2] : 21

a[3] : 6

a[4] : 89

a[5] : 23

a[6] : 99

PASS- 1

SUBLIST- 1 : 78 21 6 23

SUBLIST- 2 : 89

SUBLIST- 3 : 99

23 6 21 78 89 99

PASS- 2

SUBLIST- 1 : 21 6 23

SUBLIST- 2 : 78

SUBLIST- 3 : 89

SUBLIST- 4 : 99

21 6 23 78 89 99

PASS- 3

SUBLIST- 1 : 21 6

SUBLIST- 2 : 23

SUBLIST- 3 : 78

SUBLIST- 4 : 89

SUBLIST- 5 : 99

6 21 23 78 89 99

SORTED ELEMENTS ARE :

6 21 23 78 89 99

**ASSIGNMENT – 7**

**Program Statement**

Implementing radix sort with suitable elements

.

**Solution**

**Theory**

Here we use 1D and 2D array as the data structure. We sort the elements of the array a by the Radix sort.

**Algorithm**

radix\_sort(e,n)

Begin

creating a 2D array of n\*10 size and named it ‘r’

crate an array of size 10 named ‘top’

find the maximum number and set his digit number to the ‘len’

i🡨0

while i<len then

do

j🡨0

while j<10 then

do

top[j]=-1

j🡨j+1

end while

j🡨0

while j<n then

do

pos 🡨(e[j]/10^i) mod 10

r[pos][++top[pos]]=e[j]

j🡨j+1

end while

j🡨0

p🡨-1

while j🡨1o then

do

k🡨0

while k<=top[j] then

do

e[++p] = r[j][k]

k🡨k+1

end while

j🡨j+1

end while

i🡨i+1

end while

return e

End

**Source Code**

#include<stdio.h>

#include<math.h>

int main()

{

int r[20][10],e[20],len,i,j,k,n,p,pos,top[10];

printf("Enter the no. of Elements:");

scanf("%d",&n);

printf("Enter the highest length of Element:");

scanf("%d",&len);

printf("Enter the Element for sorting:-\n");

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

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

// Radix Sort technic begins

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

{

// Initialize the stack

for(j=0;j<10;j++)

top[j]=-1;

// Separate digits from numbers and put into stack

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

{

pos=(int)(e[j]/pow(10,i))%10;

r[pos][++top[pos]]=e[j];

}

// Collect elements from stack to the array

for(j=0,p=-1;j<10;j++)

for(k=0;k<=top[j];k++)

e[++p]=r[j][k];

}

printf("Elements is Sorted order:- ");

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

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

return 0;

}

**Output**

susovan@susovan:~/Desktop/C Programs/Assignment$ gcc -lm sort\_radix.c

susovan@susovan:~/Desktop/C Programs/Assignment$ ./a.out

Enter the no. of Elements:10

Enter the highest length of Element:4

Enter the Element for sorting:-

345

352

1278

8703

4578

3466

3256

5687

2346

2345

Elements is Sorted order:- 345 352 1278 2345 2346 3256 3466 4578 5687 8703 susovan@susovan:~/Desktop/C Programs/Assignment$

**ASSIGNMENT – 8**

**Problem Statement**

Implement Heap sort with suitable elements

**Solution**

**Theory**

Here we use array as a data structure.to implement the heap sort we first create the heap tree. So the elements are here inserted and create the max heap tree after that we sort them.

**Algorithm**

create\_heap(a,n,ite)

Begin

ptr🡨n

while ptr not equal to 1 then

do

par🡨ptr/2

if ite > a[par]

then

a[ptr] 🡨 a[par]

else

break

end if

ptr🡨par

end while

a[ptr]🡨ite

End

heap\_sort(a,n)

Begin

item 🡨 a[n]

a[n] 🡨 a[1]

n 🡨 n-1

ptr 🡨 1

while 2\*ptr+1<=n then

do

if item>a[2\*ptr] and item>a[2\*temp+1]

then

break

end if

if a[2\*ptr]>a[2\*ptr+1]

then

a[ptr]🡨a[2\*ptr]

ptr🡨ptr\*2

else

a[ptr]🡨a[2\*ptr+1]

ptr 🡨 2\*ptr +1

end if

end while

if 2\*ptr > n

then

a[ptr]🡨item

else

if a[2\*ptr]<item

then

a[ptr]🡨item

else

a[ptr]🡨a[ptr\*2]

a[ptr\*2]🡨item

end if

end if

End

**Source Code**

#include<stdio.h>

#include <stdlib.h>

void create\_heap(int a[],int n,int ite)

{

int ptr,par;

ptr=n;

while(ptr!=1)

{

par=ptr/2;

if(ite>a[par])

a[ptr]=a[par];

else

break;

ptr=par;

}

a[ptr]=ite;

}

void heap\_sort(int a[],int n)

{

int item,ptr;

item=a[n];

a[n]=a[1];

n=n-1;

ptr=1;

while(2\*ptr+1<=n)

{

if(item>a[2\*ptr]&&item>a[2\*ptr+1])

break;

if(a[2\*ptr]>a[2\*ptr+1])

{

a[ptr]=a[2\*ptr];

ptr=ptr\*2;

}

else

{

a[ptr]=a[2\*ptr+1];

ptr=2\*ptr +1;

}

}

if(2\*ptr>n)

a[ptr]=item;

else

if(a[2\*ptr]<item)

a[ptr]=item;

else

{

a[ptr]=a[ptr\*2];

a[ptr\*2]=item;

}

}

int main()

{

int a[100],n,i=1,item;

printf("Enter the total number of elements=");

scanf("%d",&n);

printf("Enter the element=");

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

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

{

printf("Enter the element=");

scanf("%d",&item);

create\_heap(a,i,item);

}

printf("The elements are in MAX Heap Tree: ");

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

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

for(i=n;i>=2;i--)

heap\_sort(a,i);

printf("\nElements after sort : ");

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

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

return 0;

}

**Output**

susovan@susovan:~/Desktop/C Programs/Assignment$ gcc sort\_heap.c

susovan@susovan:~/Desktop/C Programs/Assignment$ ./a.out

Enter the total number of elements=10

Enter the element=4

Enter the element=5

Enter the element=3

Enter the element=9

Enter the element=2

Enter the element=9

Enter the element=435

Enter the element=5

Enter the element=8

Enter the element=2

The elements are in MAX Heap Tree: 435 8 9 5 2 3 9 4 5 2

Elements after sort : 2 2 3 4 5 5 8 9 9 435

susovan@susovan:~/Desktop/C Programs/Assignment$

**ASSIGNMENT – 9**

**Problem Statement**

Write a menu driven program to implement insertion sort, selection sort, bubble sort.

**Solution**

**Theory**

Here we use array as data structure. We implement insertion, selection, bubble sort to the array for sort.

**Algorithm**

bubble\_sort(a,n)

Begin

i🡨0

while i<n-1 then

do

j🡨0

while j<n-i-1 then

do

if a[j] > a[j+1]

then

swapping the value of a[j] with a[j+1]

end if

j🡨j+1

end while

i🡨i+1

end while

End

Selection\_sort(a,n)

Begin

i🡨0

while i<n then

do

j🡨i+1

while j<n then

do

if a[i] >a[j]

then

swapping the value of a[i] with a[j]

end if

j🡨j+1

end while

i🡨i+1

end while

End

insertion\_sort(a,n)

Begin

i🡨1

while i<n then

do

j🡨i

while j>0 and a[j-1]>a[j] then

do

temp 🡨 a[j]

a[j] 🡨 a[j-1]

a[j-1] 🡨 temp

j🡨 j-1

end while

i🡨i+1

end while

End

**Source Code**

#include <stdio.h>

#include<stdlib.h>

void bouble\_sort(int a[],int n)

{

int i,j,temp;

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

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

if(a[j]>a[j+1])

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

void selection\_sort(int a[],int n)

{

int i,j,temp;

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

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

if(a[i]>a[j])

{

temp=a[j];

a[j]=a[i];

a[i]=temp;

}

}

void insertion\_sort(int a[],int n)

{

int i,j,temp;

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

{

j=i;

while(j>0 && a[j-1]>a[j])

{

temp = a[j];

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

a[j-1] = temp;

j--;

}

}

}

void display(int a[],int n)

{

int i;

printf("\nElements are :- \n");

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

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

}

int main()

{

int \*a,len,i,n;

printf("Enter the lenth of the Array = ");

scanf("%d",&len);

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

printf("\nEnter the Elements :-\n");

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

{

printf("\nA[%d] = ",i+1);

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

}

display(a,len);

printf("\n1 for bouble sort.\n2 for selection sort.\n3 for insertion sort.\nEnter Your choice = ");

scanf("%d",&n);

switch(n)

{

case 1:bouble\_sort(a,len);

break;

case 2:selection\_sort(a,len);

break;

case 3:insertion\_sort(a,len);

break;

case 4:exit(0);

}

printf("\nAfter Sorting :- ");

display(a,len);

}

**Output**

susovan@susovan:~/Desktop/C Programs/Assignment$ gcc sort.c

susovan@susovan:~/Desktop/C Programs/Assignment$ ./a.out

Enter the lenth of the Array = 10

Enter the Elements :-

A[1] = 54

A[2] = 43

A[3] = 86

A[4] = 9

A[5] = 253

A[6] = 987

A[7] = 34

A[8] = 76

A[9] = 43

A[10] = 76

Elements are :-

54

43

86

9

253

987

34

76

43

76

1 for bouble sort.

2 for selection sort.

3 for insertion sort.

Enter Your choice = 1

After Sorting :-

Elements are :-

9

34

43

43

54

76

76

86

253

987

susovan@susovan:~/Desktop/C Programs/Assignment$ ./a.out

Enter the lenth of the Array = 10

Enter the Elements :-

A[1] = 4

A[2] = 6

A[3] = 43

A[4] = 87

A[5] = 23

A[6] = 98

A[7] = 439

A[8] = 8

A[9] = 2

A[10] = 9

Elements are :-

4

6

43

87

23

98

439

8

2

9

1 for bouble sort.

2 for selection sort.

3 for insertion sort.

Enter Your choice = 2

After Sorting :-

Elements are :-

2

4

6

8

9

23

43

87

98

439

susovan@susovan:~/Desktop/C Programs/Assignment$ ./a.out

Enter the lenth of the Array = 12

Enter the Elements :-

A[1] = 546

A[2] = 43

A[3] = 98

A[4] = 8

A[5] = 23

A[6] = 987

A[7] = 453

A[8] = 657

A[9] = 4

A[10] = 87

A[11] = 76

A[12] = 23

Elements are :-

546

43

98

8

23

987

453

657

4

87

76

23

1 for bouble sort.

2 for selection sort.

3 for insertion sort.

Enter Your choice = 3

After Sorting :-

Elements are :-

4

8

23

23

43

76

87

98

453

546

657

987

susovan@susovan:~/Desktop/C Programs/Assignment$

**ASSIGNMENT – 10**

**Problem Statement**

Write a menu driven program to implement a binary search tree with different type of operation: insertion, deletion, traverse

**Solution**

**Theory**

Here we use double linked list as data structure. Every node has 3 parts one data past called x and other two are the address part right is the right children address and left is left children address. In BST we inset the child node as if it is grater then the parent node then it goes to the right side or if it is less than or equal then it goes to the left side.

**Algorithm**

create\_bst()

Begin

root🡨null

print “Enter the root Element”

input x

create a node of data item x and save its address into the root

print “Enter the elements (0 for node)”

input x

while x not equal to 0 then

do

p1🡨p2🡨root

while p2 not equal to Null then

do

p1🡨p2

if x[p2] > x

then

p2 🡨left[p2]

else

p2🡨right[p2]

end if

end while

if x[p1] > x

then

left[p1] 🡨 new node of data item x

else

right[p1] 🡨 new node of data item x

end if

print “Enter the elements (0 for node)”

input x

end while

End

inorder(p)

Begin

if p not equal to Null

then

inorder(left[p])

print x[p]

inorder(right[p])

end if

End

preorder(p)

Begin

if p not equal to null

then

print x[p]

preorder(left[p])

preorder(right[p])

end if

End

postorder(p)

Begin

if p not equal to null

then

postorder(left[p])

postorder(right[p])

print x[p]

end if

End

deletion()

Begin

find the element which have to delete

if it is not found then print “not found” and return

else

check that node is a leaf node or not

if it is a leaf node then delete it and return

else

if it has child only on one side

then

put that child tree to its positon and delete that node

else

find its predecessor

cut it from there and put from that location

End

**Source Code**

#include<stdio.h>

#include<malloc.h>

#include<stdlib.h>

typedef struct node

{

int x;

struct node \*left,\*right;

}tree;

tree \*root;

tree\* getnode(int x)

{

tree \*p;

p=(tree\*)malloc(sizeof(tree));

p->x=x;

p->left=p->right=NULL;

return(p);

}

void create\_bst()

{

tree \*p1,\*p2;

int x;

root=NULL;

printf("Enter the root Element =");

scanf("%d",&x);

root=getnode(x);

printf("Enter the Element(0 for None) =");

scanf("%d",&x);

while(x!=0)

{

p1=p2=root;

while(p2!=NULL)

{

p1=p2;

if(p2->x>x)

p2=p2->left;

else

p2=p2->right;

}

if(p1->x>x)

p1->left=getnode(x);

else

p1->right=getnode(x);

printf("Enter the Element(0 for None) =");

scanf("%d",&x);

}

}

tree \*inorder(tree \*p)

{

if(p!=NULL)

{

inorder(p->left);

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

inorder(p->right);

}

}

tree \*preorder(tree \*p)

{

if(p!=NULL)

{

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

preorder(p->left);

preorder(p->right);

}

}

tree \*postorder(tree \*p)

{

if(p!=NULL)

{

postorder(p->left);

postorder(p->right);

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

}

}

void insert()

{

tree \*p1,\*p2;

int x;

if(root == NULL)

{

printf("\nBST is Not Exist....");

return;

}

printf("Enter the Element =");

scanf("%d",&x);

p1=p2=root;

while(p2!=NULL)

{

p1=p2;

if(p2->x>x)

p2=p2->left;

else

p2=p2->right;

}

if(p1->x>x)

p1->left=getnode(x);

else

p1->right=getnode(x);

}

void del()

{

tree \*p1,\*p2,\*p3,\*p4;

int x;

printf("Enter the Element to Delete =");

scanf("%d",&x);

p1=p2=root;

while(p2!=NULL&&p2->x!=x)

{

p1=p2;

if(p2->x>x)

p2=p2->left;

else

p2=p2->right;

}

if(p2==NULL)

{

printf("Element Not Found.");

return;

}

if(p2->left==NULL&&p2->right==NULL)

{

if(p1==p2)

{

root=NULL;

free(p2);

return;

}

if(p1->left==p2)

p1->left=NULL;

else

p1->right=NULL;

}

else

if(p2->left==NULL)

{

if(p1->left==p2)

p1->left=p2->right;

else

p1->right=p2->right;

}

else

if(p2->right==NULL)

{

if(p1->left==p2)

p1->left=p2->left;

else

p1->right=p2->right;

}

else

{

p3=p4=p2->right;

while(p4->left!=NULL)

{

p3=p4;

p4=p4->left;

}

if(p3==p4)

p4->left=p2->left;

else

{

p3->left=p4->right;

p4->left=p2->left;

p4->right=p2->right;

}

if(p1->left==p2)

p1->left=p4;

else

if(p1->right==p2)

p1->right=p4;

if(p1==p2)

root=p4;

}

free(p2);

}

int main()

{

int ch;

do

{

printf("\n\n1 for Create a BST.\n2 for Inorder Display.\n3 for Preorder Display.");

printf("\n4 for Postorder Display.\n5 for Insert a element from BST.");

printf("\n6 for Delete a element from BST.");

printf("\n0 for Exit.\n\nEnter Your choice : ");

scanf("%d",&ch);

switch(ch)

{

case 1:create\_bst();

break;

case 2:printf("\nElements in INORDER : ");

inorder(root);

break;

case 3:printf("\nElements in PREORDER : ");

preorder(root);

break;

case 4:printf("\nElements in POSTORDER : ");

postorder(root);

break;

case 5:insert();

break;

case 6:del();

break;

case 0:exit(0);

}

}while(1);

return 0;

}

**Output**

susovan@susovan:~/Desktop/C Programs/Assignment$ gcc tree\_bst.c

susovan@susovan:~/Desktop/C Programs/Assignment$ ./a.out

1 for Create a BST.

2 for Inorder Display.

3 for Preorder Display.

4 for Postorder Display.

5 for Insert a element from BST.

6 for Delete a element from BST.

0 for Exit.

Enter Your choice : 1

Enter the root Element =45

Enter the Element(0 for None) =3

Enter the Element(0 for None) =6

Enter the Element(0 for None) =46

Enter the Element(0 for None) =34

Enter the Element(0 for None) =56

Enter the Element(0 for None) =23

Enter the Element(0 for None) =76

Enter the Element(0 for None) =45

Enter the Element(0 for None) =47

Enter the Element(0 for None) =7

Enter the Element(0 for None) =0

1 for Create a BST.

2 for Inorder Display.

3 for Preorder Display.

4 for Postorder Display.

5 for Insert a element from BST.

6 for Delete a element from BST.

0 for Exit.

Enter Your choice : 2

Elements in INORDER : 3 6 7 23 34 45 45 46 47 56 76

1 for Create a BST.

2 for Inorder Display.

3 for Preorder Display.

4 for Postorder Display.

5 for Insert a element from BST.

6 for Delete a element from BST.

0 for Exit.

Enter Your choice : 3

Elements in PREORDER : 45 3 6 34 23 7 46 45 56 47 76

1 for Create a BST.

2 for Inorder Display.

3 for Preorder Display.

4 for Postorder Display.

5 for Insert a element from BST.

6 for Delete a element from BST.

0 for Exit.

Enter Your choice : 6

Enter the Element to Delete =76

1 for Create a BST.

2 for Inorder Display.

3 for Preorder Display.

4 for Postorder Display.

5 for Insert a element from BST.

6 for Delete a element from BST.

0 for Exit.

Enter Your choice : 5

Enter the Element =56

1 for Create a BST.

2 for Inorder Display.

3 for Preorder Display.

4 for Postorder Display.

5 for Insert a element from BST.

6 for Delete a element from BST.

0 for Exit.

Enter Your choice : 2

Elements in INORDER : 3 6 7 23 34 45 45 46 47 56 56

1 for Create a BST.

2 for Inorder Display.

3 for Preorder Display.

4 for Postorder Display.

5 for Insert a element from BST.

6 for Delete a element from BST.

0 for Exit.

Enter Your choice : 3

Elements in PREORDER : 45 3 6 34 23 7 46 45 56 47 56

1 for Create a BST.

2 for Inorder Display.

3 for Preorder Display.

4 for Postorder Display.

5 for Insert a element from BST.

6 for Delete a element from BST.

0 for Exit.

Enter Your choice : 6

Enter the Element to Delete =45

1 for Create a BST.

2 for Inorder Display.

3 for Preorder Display.

4 for Postorder Display.

5 for Insert a element from BST.

6 for Delete a element from BST.

0 for Exit.

Enter Your choice : 2

Elements in INORDER : 3 6 7 23 34 45 46 47 56 56

1 for Create a BST.

2 for Inorder Display.

3 for Preorder Display.

4 for Postorder Display.

5 for Insert a element from BST.

6 for Delete a element from BST.

0 for Exit.

Enter Your choice : 3

Elements in PREORDER : 45 3 6 34 23 7 46 56 47 56

1 for Create a BST.

2 for Inorder Display.

3 for Preorder Display.

4 for Postorder Display.

5 for Insert a element from BST.

6 for Delete a element from BST.

0 for Exit.

Enter Your choice : 6

Enter the Element to Delete =45

1 for Create a BST.

2 for Inorder Display.

3 for Preorder Display.

4 for Postorder Display.

5 for Insert a element from BST.

6 for Delete a element from BST.

0 for Exit.

Enter Your choice : 2

Elements in INORDER : 3 6 7 23 34 46 47 56 56

1 for Create a BST.

2 for Inorder Display.

3 for Preorder Display.

4 for Postorder Display.

5 for Insert a element from BST.

6 for Delete a element from BST.

0 for Exit.

Enter Your choice : 3

Elements in PREORDER : 46 3 6 34 23 7 56 47 56

1 for Create a BST.

2 for Inorder Display.

3 for Preorder Display.

4 for Postorder Display.

5 for Insert a element from BST.

6 for Delete a element from BST.

0 for Exit.

Enter Your choice : 0

susovan@susovan:~/Desktop/C Programs/Assignment$

**ASSIGNMENT – 11**

**Problem Statement**

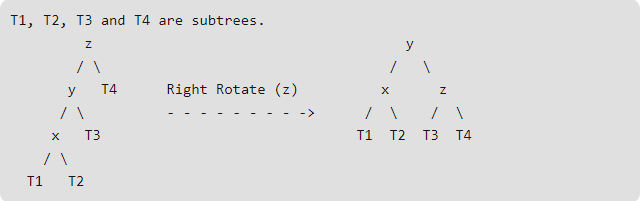
Write a menu driven program to implement a AVL tree with different type of operation: insertion, deletion, traverse

**Solution**

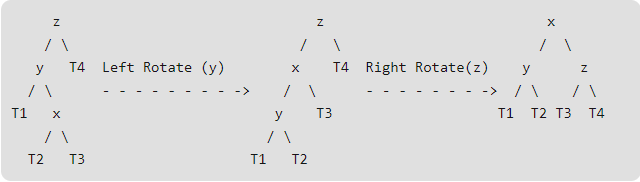
**Theory**

Here we use double linked list as data structure. Every node has 4 parts one data past called x and other two are the address part right is the right children address and left is left children address and the last part is balance factor called bf. In AVL we inset the child node as if it is grater then the parent node then it goes to the right side or if it is less than or equal then it goes to the left side but every time we check that tree is balnced or not.if it is not balanced then we done some rotations. We can find tree is balanced or not by the seeing the balancing factor. This rotations are

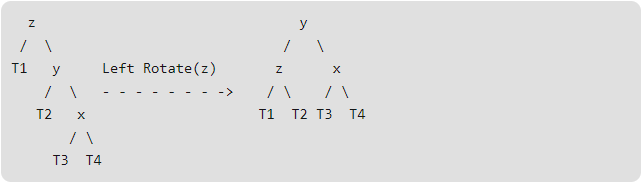
LL Rotation



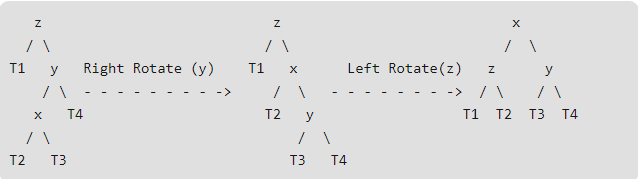
LR Rotation



RR Rotation



RL Rotation



**Algorithm**

height(p)

Begin

if p=null

then

retuen 0

end if

return 1 + max of height(left[p]) and height(right[p])

End

balnce\_bf(p)

Begin

if p not equal to null

then

balance\_bf(left[p])

bf[p] 🡨 height(left[p]) – height(right[p])

balnce\_bf(right[p])

end if

End

balance\_tree()

Begin

if bf[root] is into -1 to 1

then

tree is already balanced so return

else

tree is not balanced  
 find the point of unbalancing and balanced using rotations

end if

End

create\_AVL()

Begin

root🡨null

print “Enter the root Element”

input x

create a node of data item x and save its address into the root

print “Enter the elements (0 for node)”

input x

while x not equal to 0 then

do

p1🡨p2🡨root

while p2 not equal to Null then

do

p1🡨p2

if x[p2] > x

then

p2 🡨left[p2]

bf[p2]🡨bf[p2]+1

else

p2🡨right[p2]

bf[p2]🡨bf[p2]-1

end if

end while

if x[p1] > x

then

left[p1] 🡨 new node of data item x

else

right[p1] 🡨 new node of data item x

end if

balance\_tree()

print “Enter the elements (0 for node)”

input x

end while

End

inorder(p)

Begin

if p not equal to Null

then

inorder(left[p])

print x[p]

inorder(right[p])

end if

End

preorder(p)

Begin

if p not equal to null

then

print x[p]

preorder(left[p])

preorder(right[p])

end if

End

postorder(p)

Begin

if p not equal to null

then

postorder(left[p])

postorder(right[p])

print x[p]

end if

End

deletion()

Begin

find the element which have to delete

if it is not found then print “not found” and return

else

check that node is a leaf node or not

if it is a leaf node then delete it and return

else

if it has child only on one side

then

put that child tree to its positon and delete that node

else

find its predecessor

cut it from there and put from that location

end if

end if

end if

balance\_bf(root)

balance\_tree()

End

**Source Code**

#include<stdio.h>

#include<malloc.h>

#include<stdlib.h>

typedef struct node

{

int x;

int bf;

struct node \*left,\*right;

}tree;

tree \*root;

tree\* getnode(int x)

{

tree \*p;

p=(tree\*)malloc(sizeof(tree));

p->x=x;

p->bf=0;

p->left=p->right=NULL;

return(p);

}

int max(int a,int b)

{

if(a>b)

return a;

else

return b;

}

int height(tree \*p)

{

if(p==NULL)

return 0;

return 1+max(height(p->left),height(p->right));

}

void balance\_bf(tree \*p)

{

if(p != NULL)

{

balance\_bf(p->left);

p->bf = height(p->left) - height(p->right);

balance\_bf(p->right);

}

}

void balance\_tree()

{

tree \*p1,\*p2,\*p3;

int flag =0;

if(root->bf>-2 && root->bf<2)

return;

p1=p2=root;

while( p2->bf>1 || p2->bf<-1 )

{

p3=p1;

p1=p2;

if(p2->bf>1)

p2=p2->left;

else

p2=p2->right;

}

if(p1==p3)

flag = 1;

if(p1->bf>1)

{

if(p2->bf>0) // LL rotation

{

p3->left=p2;

p1->left=p2->right;

p2->right=p1;

if(flag) // at root node time

root = p2;

}

else // LR rotation

{

if(flag) // at root node time

{

root=p2->right;

p1->left=root->right;

root->right=p1;

p2->right=root->left;

root->left=p2;

}

else

{

p3->left=p2->right;

p1->left=p2->right->right;

p2->right->right=p1;

p2->right=p2->right->left;

p3->left->left=p2;

}

}

}

else

{

if(p2->bf<0)

{

p3->right=p2;

p1->right=p2->left;

p2->left=p1;

if(flag)

root = p2;

}

else

{

if(flag)

{

root=p2->left;

p1->right=root->left;

root->left=p1;

p2->left=root->right;

root->right=p2;

}

else

{

p3->right=p2->left;

p1->right=p2->left->left;

p2->left->left=p1;

p2->left=p2->left->right;

p3->right->right=p2;

}

}

}

balance\_bf(root);

}

tree \*inorder(tree \*p)

{

if(p!=NULL)

{

inorder(p->left);

printf("%d(%d) ",p->x,p->bf);

inorder(p->right);

}

}

tree \*preorder(tree \*p)

{

if(p!=NULL)

{

printf("%d(%d) ",p->x,p->bf);

preorder(p->left);

preorder(p->right);

}

}

tree \*postorder(tree \*p)

{

if(p!=NULL)

{

postorder(p->left);

postorder(p->right);

printf("%d(%d) ",p->x,p->bf);

}

}

void create\_avl()

{

tree \*p1,\*p2;

int x;

root=NULL;

printf("Enter the root Element =");

scanf("%d",&x);

root=getnode(x);

printf("Enter the Element(0 for None) =");

scanf("%d",&x);

while(x!=0)

{

p1=p2=root;

while(p2!=NULL)

{

p1=p2;

if(p2->x>x)

{

p2->bf=p2->bf+1;

p2=p2->left;

}

else

{

p2->bf=p2->bf-1;

p2=p2->right;

}

}

if(p1->x>x)

p1->left=getnode(x);

else

p1->right=getnode(x);

printf("\nTree before balance(preorder)\n");

preorder(root);

balance\_tree();

printf("\nTree after balance(preorder)\n");

preorder(root);

printf("\n\nEnter the Element(0 for None) =");

scanf("%d",&x);

}

}

void insert()

{

tree \*p1,\*p2;

int x;

if(root == NULL)

{

printf("\nBST is Not Exist....");

return;

}

printf("Enter the Element =");

scanf("%d",&x);

p1=p2=root;

while(p2!=NULL)

{

p1=p2;

if(p2->x>x)

{

p2->bf=p2->bf+1;

p2=p2->left;

}

else

{

p2->bf=p2->bf-1;

p2=p2->right;

}

}

if(p1->x>x)

p1->left=getnode(x);

else

p1->right=getnode(x);

printf("\nTree before balance(preorder)\n");

preorder(root);

balance\_tree();

printf("\nTree after balance(preorder)\n");

preorder(root);

}

void del()

{

tree \*p1,\*p2,\*p3,\*p4;

int x;

printf("Enter the Element to Delete =");

scanf("%d",&x);

p1=p2=root;

while(p2!=NULL&&p2->x!=x)

{

p1=p2;

if(p2->x>x)

p2=p2->left;

else

p2=p2->right;

}

if(p2==NULL)

{

printf("Element Not Found.");

return;

}

if(p2->left==NULL&&p2->right==NULL)

{

if(p1==p2)

{

root=NULL;

free(p2);

return;

}

if(p1->left==p2)

p1->left=NULL;

else

p1->right=NULL;

}

else

if(p2->left==NULL)

{

if(p1->left==p2)

p1->left=p2->right;

else

p1->right=p2->right;

}

else

if(p2->right==NULL)

{

if(p1->left==p2)

p1->left=p2->left;

else

p1->right=p2->right;

}

else

{

p3=p4=p2->right;

while(p4->left!=NULL)

{

p3=p4;

p4=p4->left;

}

if(p3==p4)

p4->left=p2->left;

else

{

p3->left=p4->right;

p4->left=p2->left;

p4->right=p2->right;

}

if(p1->left==p2)

p1->left=p4;

else

if(p1->right==p2)

p1->right=p4;

if(p1==p2)

root=p4;

}

free(p2);

printf("\nTree before balance(preorder)\n");

preorder(root);

balance\_bf(root);

printf("\nTree after bf balance(preorder)\n");

preorder(root);

balance\_tree();

printf("\nTree after balance(preorder)\n");

preorder(root);

}

int main()

{

int ch;

do

{

printf("\n\n1 for Create a AVL Tree.");

printf("\n2 for Insert a element from AVL Tree.");

printf("\n3 for Delete a element from AVL Tree.");

printf("\n4 for Inorder Display.\n5 for Preorder Display.");

printf("\n6 for Postorder Display.");

printf("\n0 for Exit.\n\nEnter Your choice : ");

scanf("%d",&ch);

switch(ch)

{

case 1:create\_avl();

break;

case 2:insert();

break;

case 3:del();

break;

case 4:printf("\nElements in INORDER : ");

inorder(root);

break;

case 5:printf("\nElements in PREORDER : ");

preorder(root);

break;

case 6:printf("\nElements in POSTORDER : ");

postorder(root);

break;

case 0:exit(0);

}

}while(1);

return 0;

}

**Output**

susovan@susovan:~/Desktop/C Programs/Assignment$ gcc tree\_avl.c

susovan@susovan:~/Desktop/C Programs/Assignment$ ./a.out

1 for Create a AVL Tree.

2 for Insert a element from AVL Tree.

3 for Delete a element from AVL Tree.

4 for Inorder Display.

5 for Preorder Display.

6 for Postorder Display.

0 for Exit.

Enter Your choice : 1

Enter the root Element =67

Enter the Element(0 for None) =3

Tree before balance(preorder)

67(1) 3(0)

Tree after balance(preorder)

67(1) 3(0)

Enter the Element(0 for None) =4

Tree before balance(preorder)

67(2) 3(-1) 4(0)

Tree after balance(preorder)

4(0) 3(0) 67(0)

Enter the Element(0 for None) =57

Tree before balance(preorder)

4(-1) 3(0) 67(1) 57(0)

Tree after balance(preorder)

4(-1) 3(0) 67(1) 57(0)

Enter the Element(0 for None) =43

Tree before balance(preorder)

4(-2) 3(0) 67(2) 57(1) 43(0)

Tree after balance(preorder)

4(1) 57(0) 43(0) 67(0) 67(0)

Enter the Element(0 for None) =67

Tree before balance(preorder)

4(0) 57(0) 43(0) 67(-1) 67(0) 67(-1) 67(0)

Tree after balance(preorder)

4(0) 57(0) 43(0) 67(-1) 67(0) 67(-1) 67(0)

Enter the Element(0 for None) =43

Tree before balance(preorder)

4(-1) 57(0) 43(0) 67(0) 43(0) 67(0) 67(0) 43(0) 67(0)

Tree after balance(preorder)

4(-1) 57(0) 43(0) 67(0) 43(0) 67(0) 67(0) 43(0) 67(0)

Enter the Element(0 for None) =41

Tree before balance(preorder)

4(-2) 57(0) 43(0) 67(1) 43(1) 41(0) 67(0) 67(1) 43(1) 41(0) 67(0)

Tree after balance(preorder)

43(2) 4(2) 57(-1) 43(0) 67(-1) 67(0) 41(0) 67(-1) 67(0)

Enter the Element(0 for None) =45

Tree before balance(preorder)

43(1) 4(2) 57(-1) 43(0) 67(0) 45(0) 67(0) 41(0) 67(0) 45(0) 67(0)

Tree after balance(preorder)

43(1) 4(2) 57(-1) 43(0) 67(0) 45(0) 67(0) 41(0) 67(0) 45(0) 67(0)

Enter the Element(0 for None) =0

1 for Create a AVL Tree.

2 for Insert a element from AVL Tree.

3 for Delete a element from AVL Tree.

4 for Inorder Display.

5 for Preorder Display.

6 for Postorder Display.

0 for Exit.

Enter Your choice : 2

Enter the Element =59

Tree before balance(preorder)

43(0) 4(2) 57(-1) 43(0) 67(1) 45(-1) 59(0) 67(0) 41(0) 67(1) 45(-1) 59(0) 67(0)

Tree after balance(preorder)

43(0) 4(2) 57(-1) 43(0) 67(1) 45(-1) 59(0) 67(0) 41(0) 67(1) 45(-1) 59(0) 67(0)

1 for Create a AVL Tree.

2 for Insert a element from AVL Tree.

3 for Delete a element from AVL Tree.

4 for Inorder Display.

5 for Preorder Display.

6 for Postorder Display.

0 for Exit.

Enter Your choice : 4

Elements in INORDER : 43(0) 57(-1) 45(-1) 59(0) 67(1) 67(0) 4(2) 41(0) 43(0) 45(-1) 59(0) 67(1) 67(0)

1 for Create a AVL Tree.

2 for Insert a element from AVL Tree.

3 for Delete a element from AVL Tree.

4 for Inorder Display.

5 for Preorder Display.

6 for Postorder Display.

0 for Exit.

Enter Your choice : 4

Elements in INORDER : 43(0) 57(-1) 45(-1) 59(0) 67(1) 67(0) 4(2) 41(0) 43(0) 45(-1) 59(0) 67(1) 67(0)

1 for Create a AVL Tree.

2 for Insert a element from AVL Tree.

3 for Delete a element from AVL Tree.

4 for Inorder Display.

5 for Preorder Display.

6 for Postorder Display.

0 for Exit.

Enter Your choice : 3

Enter the Element to Delete =59

Tree before balance(preorder)

43(0) 4(2) 57(-1) 43(0) 67(1) 45(-1) 67(0) 41(0) 67(1) 45(-1) 67(0)

Tree after bf balance(preorder)

43(2) 4(2) 57(-1) 43(0) 67(0) 45(0) 67(0) 41(0) 67(0) 45(0) 67(0)

Tree after balance(preorder)

43(0) 67(0) 57(0) 43(0) 45(0) 4(0) 67(0) 41(0) 67(0) 57(0) 43(0) 45(0) 4(0) 67(0) 41(0)

1 for Create a AVL Tree.

2 for Insert a element from AVL Tree.

3 for Delete a element from AVL Tree.

4 for Inorder Display.

5 for Preorder Display.

6 for Postorder Display.

0 for Exit.

Enter Your choice : 0

susovan@susovan:~/Desktop/C Programs/Assignment$

**ASSIGNMENT - 12**

**Problem Statement**

Find the saddle point in a matrix.

**Solution**

**ALGORITHM**

A **simple solution** is to traverse all matrix elements one by one and check if the element is Saddle Point or not.

An **efficient solution** is based on below steps.  
Traverse all rows one by one and do following for every row i.

1. Find the minimum element of current row and store column index of the minimum element.
2. Check if the row minimum element is also maximum in its column. We use the stored column index here.
3. If yes, then saddle point else continue till end of matrix.

**SOURCE CODE**

// C program to find the saddle point coordinates in the given matrix

#include <stdio.h>

int main()

{

int i, j, k, m, n, min, max,pos[2][2];

/\* Input the size of the matrix from the user \*/

printf("Enter the square matrix order : ");

scanf("%d", &m);

int matrix[m][m];

/\* Input the matrix \*/

printf("\nInput the matrix : ");

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

{

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

{

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

}

}

printf("\nThe matrix is \n");

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

{

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

{

printf("%d ", matrix[i][j]);

}

printf("\n");

}

/\* find the saddle points in the given matrix \*/

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

{

min = matrix[i][0];

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

{

if (min >= matrix[i][j])

{

min = matrix[i][j];

pos[0][0] = i;

pos[0][1] = j;

}

}

j = pos[0][1];

max = matrix[0][j];

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

{

if (max <= matrix[k][j])

{

max = matrix[i][j];

pos[1][0] = k;

pos[1][1] = j;

}

}

/\* saddle point is the minimum of a row and maximum of the column \*/

if (min == max) {

if (pos[0][0] == pos[1][0] &&pos[0][1] == pos[1][1])

{

printf("\nSaddle point (%d, %d) : %d\n",pos[0][0], pos[0][1], max);

}

}

}

return 0;

}

**OUTPUT**

Enter the square matrix order : 3

Input the matrix : 1 2 3

4 5 6

7 8 9

The matrix is

1 2 3

4 5 6

7 8 9

Saddle point (2, 0) : 7