EX.NO:2 **ARRAY IMPLEMENTATION OF LIST ADT**

AIM:

.
To write a program for List using array implementation.

DESCRIPTION:

A linked list is a sequence of data structures, which are connected together via links. Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array.

A linked list is a sequence of data structures, which are connected together via links. Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array.

Following are the important terms to understand the concept of Linked List.

- Link − each link of a linked list can store a data called an element.
- Next − each link of a linked list contains a link to the next link called Next.
- Linked List − A Linked List contains the connection link to the first link called First.

ALGORITHM:

Step1: Create nodes first, last; next, prev and cur then set the value as NULL.
Step 2: Read the list operation type.
Step 3: If operation type is create then process the following steps.

1. Allocate memory for node cur.
2. Read data in cur's data area.
3. Assign cur node as NULL.
4. Assign first=last=cur.

Step 4: If operation type is Insert then process the following steps.

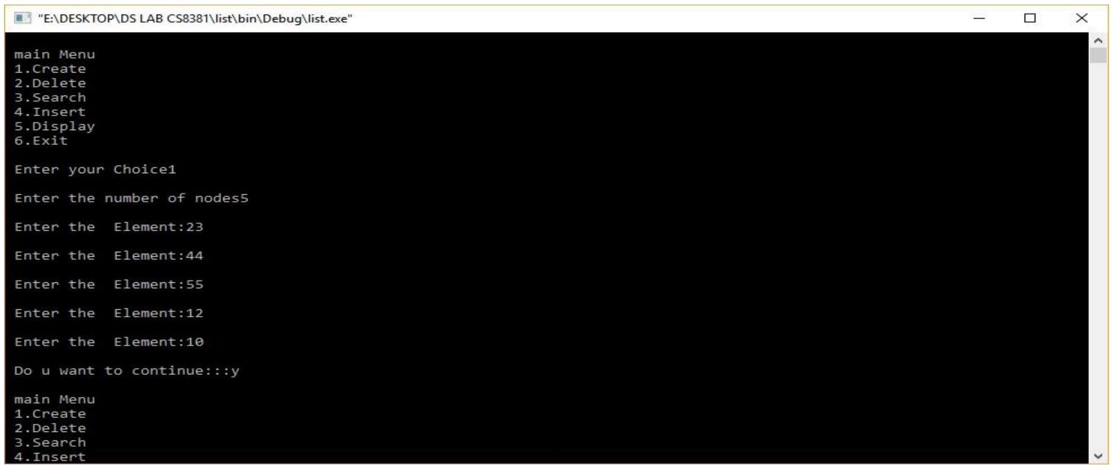
```
        1. Allocate memory for node cur.
        2. Read data in cur's data area.
        3. Read the position the Data to be insert.
        4. Availability of the position is true then assing cur's node as first and first=cur.
        5. If availability of position is false then do following steps.
            1. Assign next as cur and count as zero.
            2. Repeat the following steps until count less than postion.
                1  .Assign prev as next
                2  Next as prev of node. 3. Add count by one.

    4. If prev as NULL then display the message INVALID POSITION.
    5. If prev not qual to NULL then do the following steps.
        1. Assign cur's node as prev's node.
        2. Assign prev's node as cur.
    Step5: If operation type is delete then do the following steps.
        1. Read the position .
        2. Check list is Empty .If it is true display the message List empty.
        3. If position is first.
            1. Assign cur as first.
            2. Assign First as first of node.
            3. Reallocate the cur from memory.
        4. If position is last.
                1. Move the current node to prev.
                2. cur's node as Null.
                3. Reallocate the Last from memory.
                4. Assign last as cur.
                5. If position is enter Mediate.
            1. Move the cur to required postion.
            2. Move the Previous to cur's previous position
            3. Move the Next to cur's Next position.
            4. Now Assign previous of node as next.
            5. Reallocate the cur from memory.
step 6: If operation is traverse.
    1. Assign current as first.
    2. Repeat the following steps untill cur becomes NULL.
```

PROGRAM:

```
#include<stdio.h>
#include<conio
.h> #define
MAX 10 void
create(); void
insert(); void
deletion();
void search();
void
```

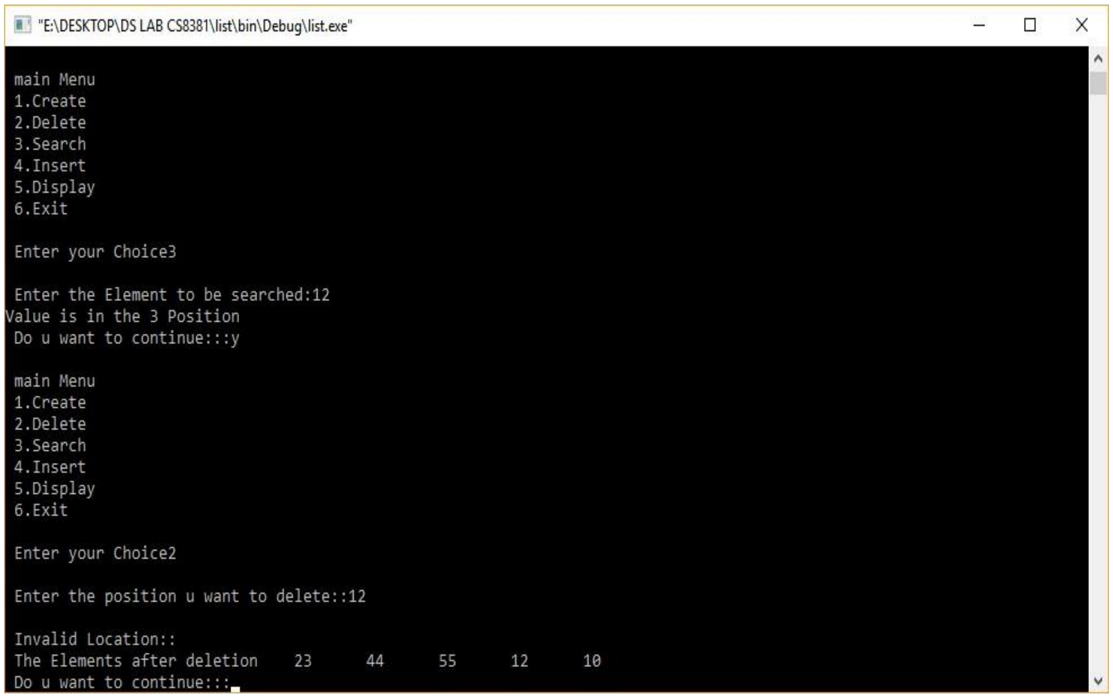
```
display(); int

a,b[20], n, p,

e, f, i, pos;

void main()

{ clrscr(); int

ch; char g='y'; do

{ printf("\n main

Menu");

printf("\n 1.Create \n 2.Delete \n 3.Search \n 4.Insert \n 5.Display\n 6.Exit
\n"); printf("\n Enter

your Choice"); scanf("%d",

&ch); switch(ch) { case 1:

create(); break; case 2:

deletion(); break; case 3:

search(); break; case 4:

insert(); break; case 5:

display(); break; case 6:

exit(); break; default:

printf("\n Enter the correct choice:");


} printf("\n Do u want to

continue:::"); scanf("\n%c", &g);

} while(g=='y'||g=='Y'); getch();

} void create() { printf("\n Enter

the number of nodes"); scanf("%d",

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

{ printf("\n Enter the

Element:",i+1); scanf("%d",

&b[i]);

} } void deletion() { printf("\n Enter the

position u want to delete::"); scanf("%d",
&pos); if(pos>=n) { printf("\n Invalid
Location::");
} else {
for(i=pos+1;i<n
;i++)
{ b[i-1]=b[i]; } n--; } printf("\n
The Elements after deletion");
for(i=0;i<n;i++)
{ printf("\t%d",
b[i]);
} } void search() { printf("\n Enter the
Element to be searched:"); scanf("%d",
&e); for(i=0;i<n;i++)
{ if(b[i]==e) { printf("Value is in
the %d Position", i);
}}} void insert() { printf("\n Enter the
position u need to insert::"); scanf("%d",
&pos); if(pos>=n)  {  printf("\n invalid
Location::");
 } else {  for(i=MAX-
1;i>=pos-1;i--)
 {  b[i+1]=b[i];  }  printf("\n Enter the
element to insert::\n");  scanf("%d",&p);
b[pos]=p;  n++;  }  printf("\n The list
after insertion::\n");
display();} void display(){ printf("\n The
Elements of The list ADT are:"); for(i=0;i<n;i++)
{ printf("\n\n%d", b[i]);
}}
```

OUTPUT



"E:\DESKTOP\DS LAB CS8381\list\bin\Debug\list.exe"

```
main Menu
1.Create
2.Delete
3.Search
4.Insert
5.Display
6.Exit

Enter your Choice1

Enter the number of nodes5

Enter the  Element:23

Enter the  Element:44

Enter the  Element:55

Enter the  Element:12

Enter the  Element:10

Do u want to continue:::y

main Menu
1.Create
2.Delete
3.Search
4.Insert
```



"E:\DESKTOP\DS LAB CS8381\list\bin\Debug\list.exe"

```
 main Menu
 1.Create
 2.Delete
 3.Search
 4.Insert
 5.Display
 6.Exit

 Enter your Choice3

 Enter the Element to be searched:12
Value is in the 3 Position
 Do u want to continue:::y

 main Menu
 1.Create
 2.Delete
 3.Search
 4.Insert
 5.Display
 6.Exit

 Enter your Choice2

 Enter the position u want to delete::12

 Invalid Location::
 The Elements after deletion    23      44      55      12      10
 Do u want to continue:::
```

RESULT:

Thus the C program for array implementation of List ADT was created, executed and output was verified successfully