|  |  |
| --- | --- |
|  | **DEPARTMENT OF COMPUTER ENGINEERING** |

**Experiment No. 11 A**

|  |  |
| --- | --- |
| Semester | S.E. Semester III – Computer Engineering |
| Subject | Data Structures Lab (CSL301) |
| Subject Professor In-charge | Prof. Swapnil S. Sonawane |
| Assisting Teachers | Prof. Swapnil S. Sonawane |

|  |  |
| --- | --- |
| Student Name | Trisha Shah |
| Roll Number | 20102A0004 |

**Title:**

Implement ADT using Linked List.

**Objective:**

Students will be able to handle various operations like searching, insertion, deletion and traversals on various data structures.

**Explanation:**

Implement a stack using single linked list concept. All the single linked list operations perform based on Stack operations LIFO(last in first out) and with the help of that knowledge we are going to implement a stack using single linked list. using single linked lists so how to implement here it is linked list means what we are storing the information in the form of nodes and we need to follow the stack rules and we need to implement using single linked list nodes so what are the rules we need to follow in the implementation of a stack a simple rule that is last in first out and all the operations we should perform so with the help of a top variable only with the help of top variables are how to insert the elements let’s see

A stack can be easily implemented through the linked list. In stack Implementation, a stack contains a top pointer, which is “head” of the stack where pushing and popping items happens at the head of the list. first node have null in link field and second node link have first node address in link field and so on and last node address in “top” pointer.

Stack Operations:

push() : Insert the element into linked list nothing but which is the top node of Stack.

pop() : Return top element from the Stack and move the top pointer to the second node of linked list or Stack.

peek(): Return the top element.

display(): Print all element of Stack.

**Program Code:**

#include<stdio.h>

#include<stdlib.h>

typedef struct node

{

int data;

struct node \*next;

}node;

typedef struct LL

{

node \*start;

}LL;

void display(LL \*l)

{

node \*p;

if(l->start==NULL)

{

printf("\nList is empty...");

}

else

{

p=l->start;

while(p!=NULL)

{

printf("\n%d",p->data);

p=p->next;

}

}

}

void push(LL \*l,int x)

{

node \*newrec;

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

newrec->data=x;

newrec->next=NULL;

if(l->start==NULL)

{

l->start=newrec;

}

else

{

newrec->next=l->start;

l->start=newrec;

}

}

void pop(LL \*l)

{

node \*p;

if(l->start==NULL)

{

printf("\nDeletion not possible...");

}

else

{

p=l->start;

l->start=l->start->next;

free(p);

}

}

int main()

{

int ch,x;

LL l;

l.start=NULL;

while(1)

{

printf("\nMenu:\n1-PUSH\n2-POP\n3-DISPLAY\n4-EXIT\nEnter Choice=");

scanf("%d",&ch);

if(ch==4)

break;

switch(ch)

{

case 1:

{

printf("\nEnter element to be inserted=");

scanf("%d",&x);

push(&l,x);

display(&l);

}

break;

case 2:

{

pop(&l);

display(&l);

}

break;

case 3:

{

display(&l);

}

break;

default:

{

printf("\nInvalid Choice...");

}

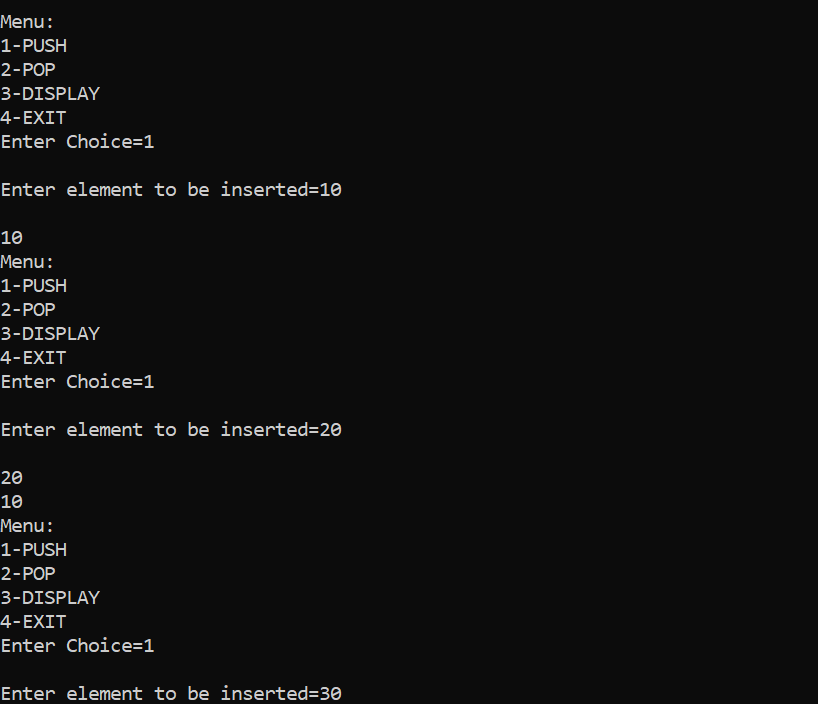
}

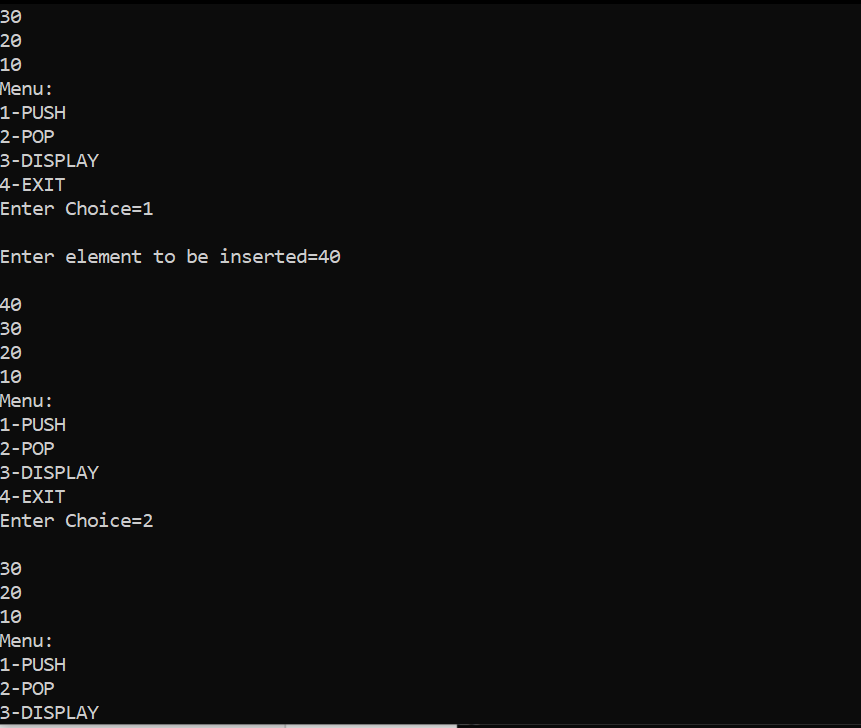
}

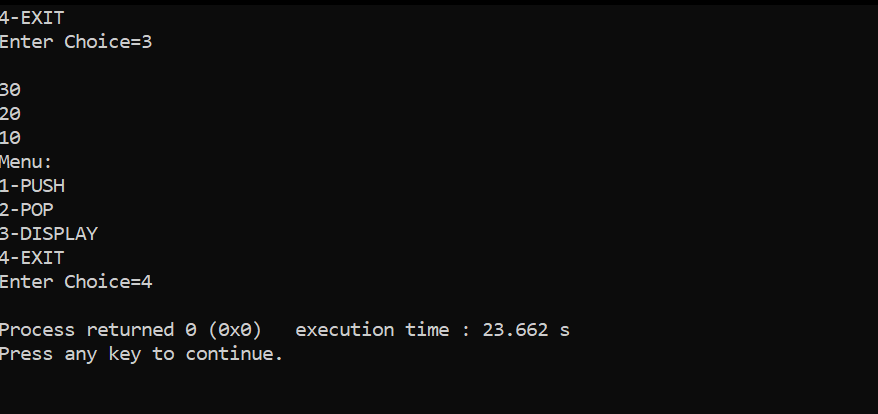
return 0;

}

**Output:**

****

****

****

**Conclusion:**

Through this experiment, students were able to successfully implement a stack using linkedlist.