**Experiment 15**

**Implementation Of Stack Using Linked List**

**Date:** 10-11-2020

**Aim:** Implementation of Stack using Linked List

**Data Structure Used:** Stack

**Operation Used:** Comparisons

**Algorithm:**

**Algorithm for Push**

**Input:** The Stack (S) implemented using Linked List, the pointer to the element at the top (TOP), ITEM to be inserted

**Output:** The Stack (S) with ITEM inserted at the top.

**Data Structure:** Stack and linked list

**Steps:**

Step 1: Start

Step 2: new = GetNode(Node)

Step 3: if(new!=NULL) then

Step 1: new→data = ITEM

Step 2: new→link = NULL

Step 3: if(Top!=NULL) then

Step 1: new→link = Top→Link

Step 4: endif

Step 5: Top = new

Step 4: else

Step 1: print(“Insertion not possible”)

Step 2: exit(1)

Step 5: endif

Step 6: Stop

**Description of the algorithm**

This algorithm places a new Node ‘new’ with the value of ITEM and the link part pointing to the previous Top element in the Stack (S) making it the new Top element

**Algorithm for Pop**

**Input:** The Stack (S) implemented using Linked List, the pointer to the element at the top (TOP)

**Output:** The Stack (S) with , ITEM to be removed and the ITEM

**Data Structure:** Stack and Linked list

**Steps:**

Step 1:Start

Step 2: if(Top == NULL)

Step 1: print(“The Stack is empty”)

Step 2: exit

Step 3: else

Step 1: ITEM = Top→data

Step 2: remove = Top

Step 3: Top= Top→link

Step 4: RetrunNode(remove)

Step 5: return ITEM

Step 4:endif

Step 5: Stop

**Description of the algorithm:**

This algorithm stores the value of the current Top item in a variable, and stores the value in a variable remove. Then it assigns Top to Top→Link and returns the remove variable to the memory.

**Program Code:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Stack Implementation using a Linked List

\* Done By: Rohit Karunakaran

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<stdio.h>

#include<stdlib.h>

typedef struct Linked\_List\_Node

{

struct Linked\_List\_Node \*link;

int data;

}Node;

typedef struct Linked\_Stack

{

Node \*Top;

}Stack;

Stack\* initStack()

{

Stack \*s = (Stack\*) malloc (sizeof(Stack));

s->Top = NULL;

return s;

}

//Insertion Algorithms

void push(Stack \*s,int val)

{

Node \*new\_node = (Node\*) malloc(sizeof(Node));

if(new\_node!=NULL)

{

new\_node->data = val;

new\_node->link = s->Top;

s->Top = new\_node;

}

else

{

printf("Stack Is Full");

exit(1);

}

return ;

}

//Deletion Algorithms

int pop(Stack \*s)

{

if(s->Top == NULL)

{

printf("Stack Is Empty");

exit(0);

return 0;

}

else

{

Node\* ptr = s->Top;

s->Top = s->Top->link;

int elem = ptr->data;

free(ptr);

return elem;

}

}

void displayStack(Stack \*s)

{

Node\* ptr = s->Top;

if(ptr!=NULL)

{

printf("The Stack is: Top -> ");

while(ptr!=NULL)

{

if(ptr==s->Top){

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

}

else{

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

}

ptr=ptr->link;

}

printf("\n");

}

else

{

printf("The Stack is empty\n");

}

}

int menu(Stack\* s)

{

int RUN = 1;

while(RUN)

{

printf("\n");

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

printf(" MENU \n");

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

printf("1.Push\n");

printf("2.Pop\n");

printf("3.Display the stack\n");

printf("4.Exit\n");

printf("Enter Choice: ");

int choice;

int elem;

scanf("%d%\*c",&choice);

switch(choice)

{

case 1: printf("Enter the element to be inserted: ");

scanf("%d%\*c",&elem);

push(s,elem);

printf("\n");

break;

case 2: elem = pop(s);

printf("The Element removed is %d",elem);

printf("\n");

break;

case 3: displayStack(s);

break;

case 4: RUN=0;

break;

default: printf("Enter a valid choice\n");

printf("\n");

break;

}

}

printf("Exiting........");

return RUN;

}

int main()

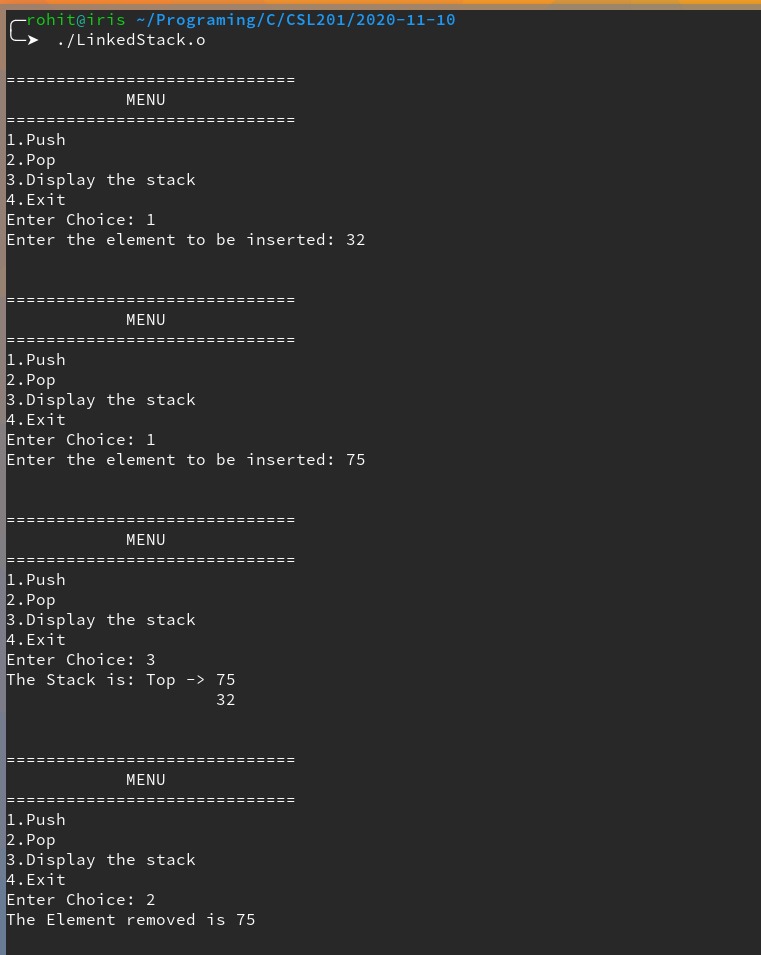
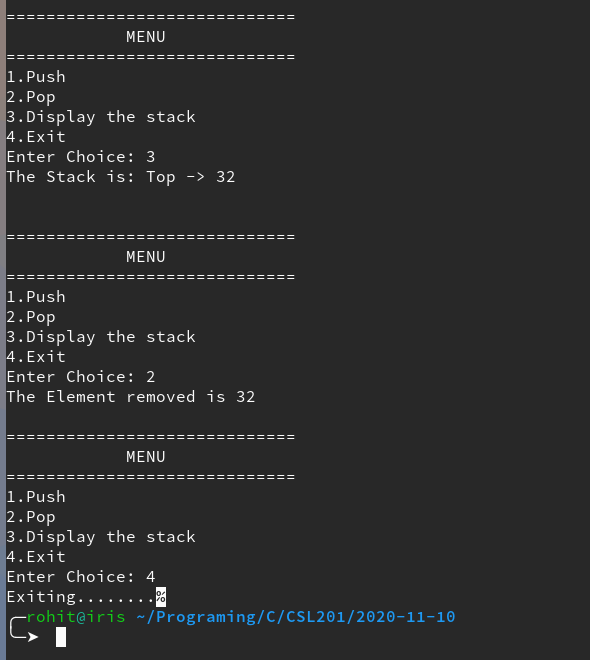
{

Stack \*s = initStack();

return menu(s);

}

**Result:** The Program is successfully compiled and the desired result is obtained

**Sample Input****/Output**