Experiment 13
Date 01-12-2022

STACK USING LINKED LIST

Aim

Write a menu driven C Program to implement a stack using linked list with the following operations:
a.Push elements to the stack.
b.Pop elements from the stack.
c.Display the stack after each operation

1 Stack using Linked List

1.1 Algorithm

Step 1: Start

Step 2: Create structure having members item and a self referential link

Step 3: Declare structure variable stack_head and dynamically allocate memory to the node

Step 4: Set default values of item and link of stack_head as NULL

Step 5: Display menu items(1.Push, 2.Pop, 3.Exit)

Step 6: Input option,op

Step 7: If op=1, then push()

Step 8: If op=2, then pop()

Step 9: If op=3, then goto STEP 13

Step 10: Else print "invalid input", goto STEP 5

Step 11: display()

Step 12: Goto STEP 5

Step 13: Stop

Function push() inserts element to top of stack

Step 14: Start of function push()

Step 15: Allocate memory for node,new using get_node()

Step 16: If new=NULL, return

Step 17: Else let (new→link)←(header→link), input new→item, let (header→link) ←new

Step 18: return

Function pop() deletes the element at top of stack

Step 19: Start of function pop()

Step 20: If (header→link)=NULL, return

Step 21: Else let (header→link)←(ptr→link)

Step 22: Return the deleted node to memory pool using return_node(ptr)

Step 23: return

Function get_node() returns a node of required size from memory pool

Step 24: Start of function get_node()

Step 25: Return a dynamically allocated node of the size of structure

Function return_node() returns the deleted node to memory pool

Step 26: Start of function return_node(pr)

Step 27: Set the value of item and link of ptr to NULL

Step 28: return

{Function display() displays elements of the stack}

Step 29: Start of function display()

Step 30: Print ptr→item from ptr←(header→link) until ptr=NULL with ptr← (ptr→link) after each iteration

Step 31: return

1.2 Program

```
#include <stdio.h>
#include <stdlib.h>
struct node{
    int data;
    struct node *next;
};
struct node *head, *top;
struct node* createNode(){
return (struct node*)malloc(sizeof(struct node));
}
void popLL()
{
struct node *ptr;
if(head->next==NULL){
printf("Stack Empty\n");
return;
}
printf("The deleted element is: %d\n",top->data);
ptr=head;
while(ptr->next!=top){
ptr=ptr->next;
}
ptr->next=NULL;
free(top);
top=ptr;
}
void pushLL(int item)
{
struct node *new;
new=createNode();
new->data=item;
new->next=NULL;
if(head->next==NULL){
top=head;
}
top->next=new;
top=new;
}
void displayLL()
{
struct node *ptr;
printf("The current stack is: ");
if(head->next==NULL){
printf("Stack Empty\n");
```

```
return;
}
ptr=head->next;
while(ptr!=NULL){
printf("%d ",ptr->data);
ptr=ptr->next;
}
}
void main()
{
int choice,num;
head= createNode();
head->next=NULL;
printf("Choose from the following:\n\t1.push an item to stack\n");
printf("\t2.Pop an item from stack\n\t3.display stack\n\t4.Exit\n");
while(1)
{
printf("\n\nEnter choice: ");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("Enter Item to push: ");
scanf("%d",&num);
pushLL(num);
displayLL();
break;
case 2: popLL();
displayLL();
break;
case 3: displayLL();
break;
case 4: break;
default: printf("Invalid Input\n");
}
if(choice == 4)
break;
}
}
```

1.3 Sample Output

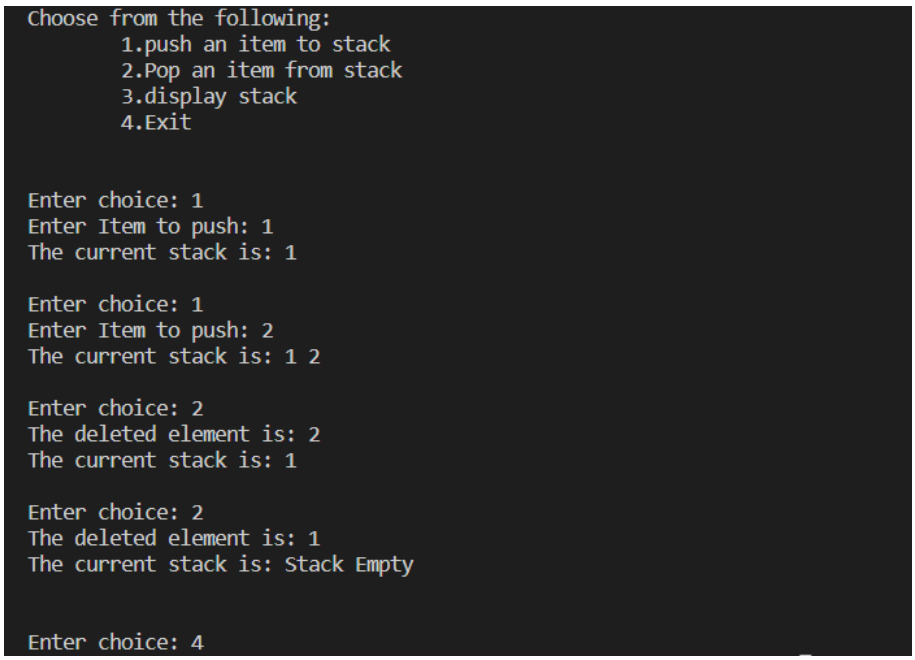

Figure 1: Input and Output

1.4 Result

A menu driven C program was made with the operations: Push, Pop and Display. Push inserts an element onto a stack in the form of a linked list, Pop deletes and item from the top of the stack and Display prints the entire contents of the stack. Stack implemented using linked list.