Data Structures Odyssey: Exploring the Foundations of Computing

Ex. No.:04	**Implementation of Stack using Array and Linked List Implementation**	**Date:21/03/2024**

Write a C program to implement a stack using Array and linked List implementation and execute the following operation on stack.

(i) Push an element into a stack

(ii) Pop an element from a stack

(iii) Return the Top most element from a stack

(iv) Display the elements in a stack

Algorithm:

```
1) Start
2) Stack using Array:
3) Define a fixed-size array to store the stack elements and initialize a variable top to -1.
  Implement functions for the following operations:
- Push: Increment top and insert the element at the top index.
- Pop: Remove the element at the top index and decrement top.
- Peek: Return the element at the top index without removing it.
- IsEmpty: Check if top is -1 to determine if the stack is empty.
- IsFull: Check if top is equal to the maximum array size to determine if the stack is full.

4) Stack using Linked List:
a. Define a Node structure with data and a pointer to the next Node.
b. Initialize a head pointer to NULL.
c. Implement functions for the following operations:
- Push: Create a new Node with the given data and insert it at the beginning of the list.
- Pop: Remove the first Node from the list.
- Peek: Return the data of the first Node without removing it.
- IsEmpty: Check if the head pointer is NULL to determine if the stack is empty. 5) Stop
```

PROGRAM: ARRAY IMPLEMENTATION

```
#include <stdio.h>
#include <string.h>
#include<ctype.h>
int top = -1; int
stack[100]; void
push (int data) {
stack[++top] = data;
} int pop () { int
data; if (top == -
1) return -1; data
= stack[top];
stack[top] = 0;
top--; return
(data);
}
int main()
{
char str[100];
int i, data = -1, operand1, operand2, result;
printf("Enter ur postfix expression:");
fgets(str, 100, stdin); for (i = 0; i <
strlen(str); i++)
 { if
(isdigit(str[i]))
{
data = (data == -1) ? 0 : data; data
= (data * 10) + (str[i] - 48);
continue;
} if (data != -
1)
{
push(data);
}
if (str[i] == '+' || str[i] == '-'|| str[i] == '*' || str[i] == '/')
{
operand2 = pop(); operand1 = pop();
if (operand1 == -1 || operand2 == -1)
break; switch (str[i])
{
case '+':
result = operand1 +
operand2; push(result);
break; case '-':
result = operand1 - operand2;
push(result); break; case '*':
result = operand1 * operand2;
push(result); break; case '/':
result = operand1 / operand2;
push(result); break;
}
}


data = -1;
} if (top ==
0)
printf("The answer is:%d\n", stack[top]); else
printf("u have given wrong postfix expression\n");
return 0;
}
```

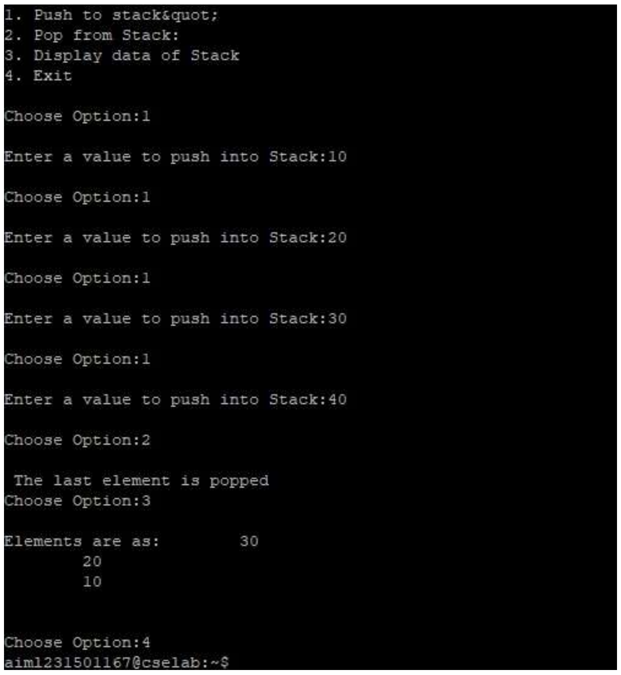
LINKED LIST IMPLEMENTATION:

```
#include <stdio.h>
#include <stdlib.h> struct
Node
```

```
{
int Data; struct
Node *next; }*top;
void popStack()
{
struct Node *temp, *var=top;
if(var==top)
{
top = top->next; free(var);
}
else printf("\nStack
Empty");
}
void push(int value)
{
struct Node *temp;
temp=(struct Node *)malloc(sizeof(struct Node)); temp->Data=value; if (top == NULL)
{
top=temp; top->next=NULL;
}

else
{
temp->next=top; top=temp;
}
}
void display()
{
struct Node *var=top;
if(var!=NULL)
{
printf("\nElements are as:");
while(var!=NULL)
{
```

```
printf("\t%d\n",var->Data);
var=var->next;
} printf("\n");
}
else printf("\nStack is
Empty&quot;");
}
int main()
{ int
i=0;
top=NULL;

printf("\n1. Push to stack&quot;");
printf("\n2. Pop from Stack:");
printf("\n3. Display data of Stack");
printf("\n4. Exit\n"); while(1)
{
printf ("\nChoose Option:");
scanf("%d",&i); switch(i)
{
case 1:
{
int value;
printf("\nEnter a value to push into Stack:");
scanf("%d",&value); push(value); break;
}
case 2:
 {
popStack(); printf("\n The last element
is popped"); break;
}
case 3:
{
display(); break;
}
case 4:
{

struct Node *temp;
while(top!=NULL)
{
temp = top->next;
free(top); top=temp;
} exit(0);
}
default:
{
printf("\nwrong choice for operation");
}}}}
```

OUTPUT:



```
1. Push to stack&quot;
2. Pop from Stack:
3. Display data of Stack
4. Exit

Choose Option:1

Enter a value to push into Stack:10

Choose Option:1

Enter a value to push into Stack:20

Choose Option:1

Enter a value to push into Stack:30

Choose Option:1

Enter a value to push into Stack:40

Choose Option:2

 The last element is popped
Choose Option:3

Elements are as:        30
        20
        10


Choose Option:4
aiml231501167@cselab:~$
```

RESULT: Thus the program was successfully executed.