Department of
Computer Science and Engineering

Title: Linked List Implementation of Stack

Data Structure Lab

CSE 106

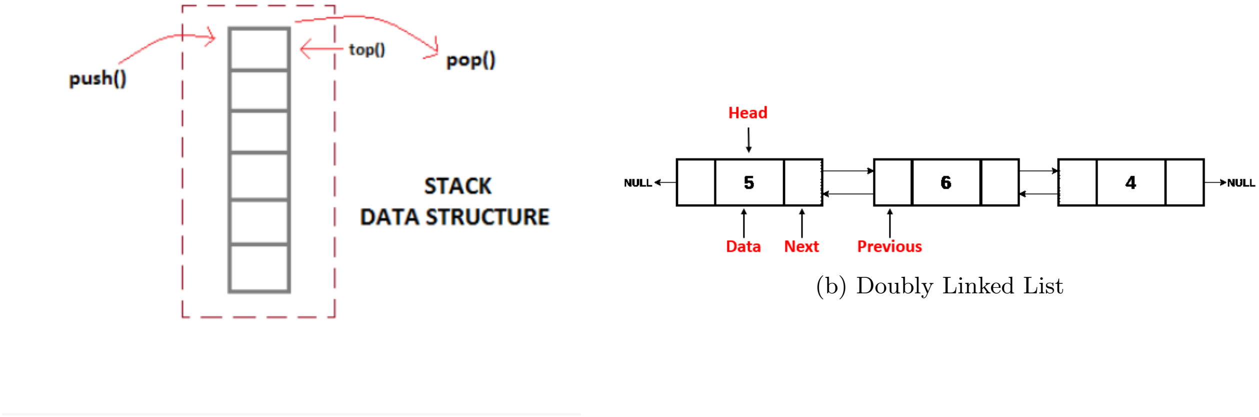
Green University of Bangladesh

1 Objective(s)

- To attain knowledge on the Stack data structure and Linked List.
- To implement Stack using Doubly Linked List.

2 Problem analysis

Stack is a linear data structure in which the insertion and deletion operations are performed at only one end. In a stack, adding and removing of elements are performed at a single position which is known as "top". That means, a new element is added at top of the stack and an element is removed from the top of the stack. In stack, the insertion and deletion operations are performed based on LIFO (Last In First Out) principle. In a stack, the insertion operation is performed using a function called "push" and deletion operation is performed using a function called "pop".
A linked list is a linear collection of data elements whose order is not given by their physical placement in memory. Instead, each element points to the next. It is a data structure consisting of a collection of nodes which together represent a sequence.
This structure allows for efficient insertion or removal of elements from any position in the sequence during iteration.
A stack can be easily implemented through the linked list.



(b) Doubly Linked List

(a) Stack Data Structure

Figure 1: Illustration of Stack and Doubly Linked List

2.1 Stack Operations using Linked List

- Inserting an element into the Stack
- Deleting an Element from a Stack
- Displaying stack of elements

3 Algorithm

Algorithm 1: Linked list implementation of stack

```
Input: Element
/* Algorithm for push operation                                     */
Step 1: [Declare the Variables]
Step 2: Read operator
if opt == 1 then
    Step 3.1: READ n
    Step 3.2: while n<n-1 do
        Step 3.2.1: READ d
        Step 3.2.2: CALL INSERT( start , d)
    end
    Step 3.3: [End of while Structure]
end
Step 4: if opt == 2 then
    Step 4.1: READ x Step 4.2: CALL del(start,x)
end
Step 5: if opt == 3 then
    Step 5.1: READ x Step 5.2: CALL FIND
end
Step 6: if opt == 4 then
    Step 6.1: READ x Step 6.2: CALL FINDPREVIOUS
end
Step 7: if opt == 5 then
    Step 7.1: READ x Step 7.2: CALL FINDNEXT(start, x)
end
Step 8: if opt == 6 then
    CALL len(Start)
end
Step 9: if opt == 7 then
    THEN CALL printlist(Start)
end
Step 10: if opt == 8 then
    CALL erase (Start)
end
Step 11: Exit
```

4 Flowchart

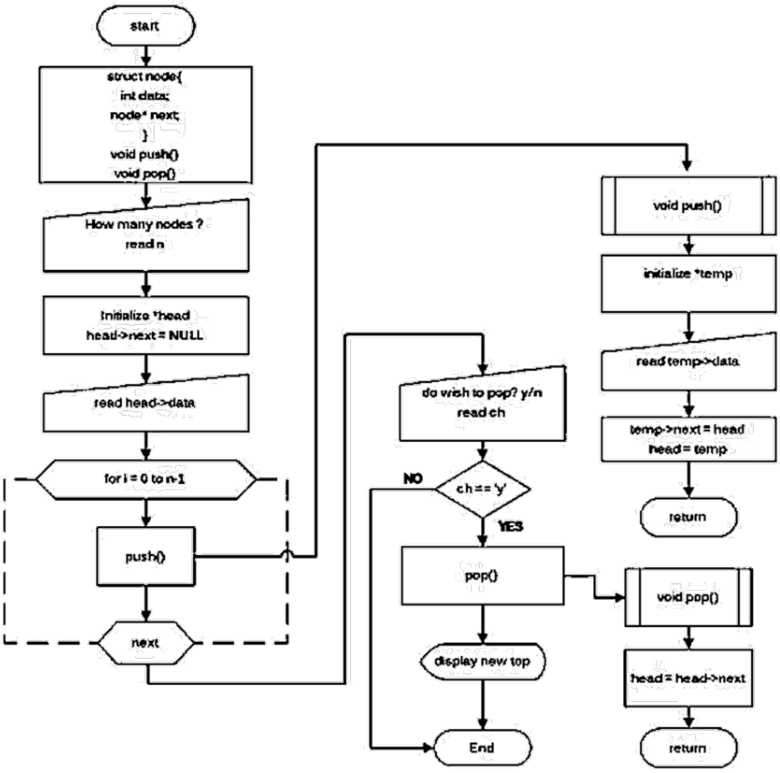


Figure 2: PUSH and POP operation with Doubly Linked List

5 Implementation in C

```c
#include <stdio.h>
#include<stdlib.h>

struct node
{
    int val;
    struct node *left;
    struct node *right;
};

struct node* getnode( )
{
    return ( struct node *)malloc( sizeof( struct node ));
};

struct node* top;
int c=0;
struct node* add( struct node *FIRST)
{
    struct node *T;
    if( FIRST == NULL )
    {
        FIRST = getnode();
        T =top=FIRST;
        T->left=NULL;
    }
    else
    {   T=getnode();
        top->right=T;
        T->left=top;
        top=T;
    }
    printf("Enter a val: ");
    c++;
    scanf("%d",&T->val );
    T->right= NULL;
    return FIRST;
}

struct node* del( struct node *FIRST)
{
    struct node *T;
    if(( FIRST== NULL)&&(c<1))
    {
        printf("Underflow\n"); return FIRST;
    }
    T=top;
    printf("The deleted value is : %d\n",T->val);
    if(c!=1)
    {
        top=T->left;
        T->left= NULL; top->right=NULL;
    }
    else if(c==1)
    {
        FIRST=NULL;
    }
    c--;
    return FIRST;
}

void display( struct node *T)
{
    if(T==NULL)
    {
        printf("Empty\n");
        return;
    }
    printf("Null -->  ");
    while( T != NULL )
    {
        printf("%d -->  ",T->val);
        T = T->right;
    }
    printf("Null \n\n");
}

int main()
{
    struct node *F;
    int ch;
    F = NULL; top=F;
    printf("DOUBLY LINKED LIST IMPLEMENTATION OF STACK\n\n");
    while( 1 )
    {
        printf("1. Add new element\n");
        printf("2. Delete element\n");
        printf("3. Display elements \n");
        printf("4. Exit \n");
        printf("Choice: ");
        scanf("%d", &ch );
        printf("\n");
        if( ch == 1 )
            F = add ( F );
        else if( ch == 2 )
            F=del(F);
        else if(ch==3)
            display( F );
        else if( ch == 4 )
            return;
    }
    return 0;
}
```

6 Input/Output (Compilation, Debugging & Testing)

Input & Output:

```
DOUBLY LINKED LIST IMPLEMENTATION OF STACK

1. Add new element
2. Delete element
3. Display elements
```

```
4. Exit
Choice: 1

Enter a val: 100
1. Add new element
2. Delete element
3. Display elements
4. Exit
Choice: 1

Enter a val: 10
1. Add new element
2. Delete element
3. Display elements
4. Exit
Choice: 1

Enter a val: 25
1. Add new element
2. Delete element
3. Display elements
4. Exit
Choice: 3

Null –> 100 –> 10 –> 25 –> Null

1. Add new element
2. Delete element
3. Display elements
4. Exit
Choice: 2

The deleted value is : 25
1. Add new element
2. Delete element
3. Display elements
4. Exit
Choice: 2

The deleted value is : 10
1. Add new element
2. Delete element
3. Display elements
4. Exit
Choice: 3

Null –> 100 –> Null

1. Add new element
2. Delete element
3. Display elements
4. Exit
Choice: 4


Process returned 4 (0x4) execution time : 51.314 s
Press any key to continue.
```

7 Discussion & Conclusion

Based on the focused objective(s) to have a clear understanding about the Linked List and all its operation related to it while implementing in stack, the additional lab exercise made me more confident towards the fulfilment of the objectives(s).

8 Lab Task (Please implement yourself and show the output to the instructor)

1. Modify the C program that is able to update as well as search any element present in the stack.
2. Implement the STACK for both numeric and character items.
3. Write appropriate code for controlling OVERFLOW and UNDERFLOW.

9 Lab Exercise (Submit as a report)

- Find the specific node of element that is present or not in the linked list.
- Call a function that will generate the size of the linked list.

10 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.