TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

PURWANCHAL CAMPUS

Dharan-8

A Lab Report On: Implementation on Singly Linked List

Submitted By

Name: Dhiraj KC
Roll No. : PUR077BEI014
Faculty: Electronics, Communication and Information
Group: A

Submitted To

Department Of Electronics and Computer Engineering
Checked By: Pukar Karki

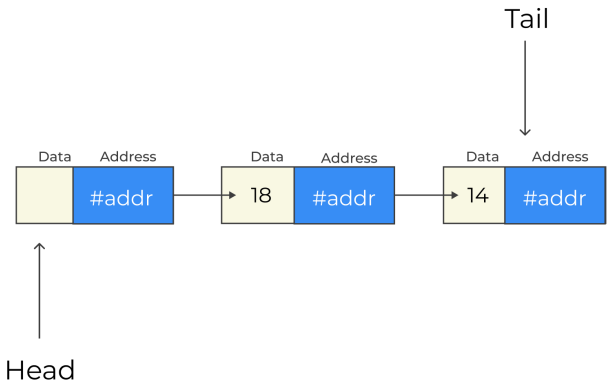
TITLE: ALGORITHM OF IMPLEMENTATION OF SINGLY LINKED LIST

THEORY

Singly Linked List in C is one of the simplest linear data structures, that we use for storing our data in an easy and efficient way. Linked List in C comprises nodes like structures, which can further be divided into 2 parts in the case of a singly linked list. These two parts are-:

1. Node – for storing the data.
2. Pointer – for storing the address of the next node.

In a Singly linked list there is only one pointer type variable, that contains the address of the next node.



ALGORITHM

1. Start
2. Read Element to be Inserted
3. Allocate Memory for NewNode
4. Set NewNode data as element and NewNode pointer to NULL
5. If first == NULL;
   First = Last = NewNode
   Display "Success"
6. Else
   NewNode next pointer to first
   First = NewNode
   Display "Success"
7. Make two temporary node as temp1 and temp2
8. If (First == NULL)
   Display "List is Empty"
   Else If (Next of First == NULL)
   temp = First
   First = NULL
   free(temp)
   Else
   temp = First
   while(temp->Next != NULL)
   temp2 = temp
   temp = temp -> next
   temp2 -> Next = NULL
   free(temp)
9. Stop

PROGRAM

```
// Implementing SLL
#include <stdio.h>
#include <stdlib.h>

struct SLL
{
 int data;
 struct SLL *next;
} *top;

void push(int element)
{
 struct SLL *NewNode;
 NewNode = (struct SLL *)malloc(sizeof(struct SLL));
 if (NewNode == NULL)
  printf("Memory Allocation Failed\n");
 else
 {
  NewNode->data = element;
  NewNode->next = NULL;
  if (top == NULL)
  {
   top = NewNode;
  }
  else
  {
   NewNode->next = top;
   top = NewNode;
  }
  printf("Success\n");
 }
}

int pop()
{
 struct SLL *temp;
 if (top == NULL)
 {
  printf("Stack Underflow\n");
  return -1;
 }
 else
 {
  int element = top->data;
  if (top->next == NULL)
  {
   temp = top;
   free(temp);
```

```
    top = NULL;
   }
   else
   {
    temp = top;
    top = top->next;
    free(temp);
   }
   return element;
  }
}

int main()
{
  top = NULL;
  int choice, element;
  do
  {
   printf("1. PUSH\n2. POP\n3. EXIT\n");
   printf("Choice: \n");
   scanf("%d", &choice);

   switch (choice)
   {
   case 1:
    printf("Enter a value: ");
    scanf("%d", &element);
    push(element);
    break;

   case 2:
    element = pop();
    printf("%d was popped", element);
    break;
   default:
    break;
   }
  } while (choice != 3);

  return 0;
}
```

OUTPUT

```
1. PUSH
2. POP
3. EXIT
Choice:
1
Enter a value: 4
Success
1. PUSH
2. POP
3. EXIT
Choice:
1
Enter a value: 5
Success
1. PUSH
2. POP
3. EXIT
Choice:
1
Enter a value: 6
Success
1. PUSH
2. POP
3. EXIT
Choice:
2
6 was popped
1. PUSH
2. POP
3. EXIT
Choice:
2
5 was popped
1. PUSH
2. POP
3. EXIT
Choice:
2
4 was popped
1. PUSH
2. POP
3. EXIT
Choice:
2
Stack Underflow
-1 was popped
1. PUSH
2. POP
3. EXIT
Choice:
3
```