ASSIGNMENT NO:9 **DATE:27/03/2015**

PROGRAM TITLE:Implement Stack using Linked list.

THEORY: Each node of a single linked list holds the address of only the next node in the linked list. The stack has only one pointer TOS that points to the top of the stack. Stack follows the LIFO(Last In First Out) logic.

PROGRAM ALGORITHM:

```
Algo_push(top,item)
{
      allocate temp node with item;
      if(inter=NULL)
      {
            print overflow; //This only occurs when the memory is full
            return;
      }
      if(top=NULL) //i.e when the stack is empty
      {
            top=inter;
      }
      else
      {
            next(inter)=top;
            top=inter;
      }
      return top;
}

Algo_pop(top)
{
      if (top=NULL)
      {
            print underflow;
            return;
      }
      else
      {
            p=top;
            top=next(top);
            free (p);
            return;
      }

}
```

PROGRAM CODE:

```
/*C Program to Implement Stack using linked list*/
#include <stdio.h>
#include <stdlib.h>
struct Node
{
```

```
      int data;
      struct Node *next;
};
typedef struct Node *NODEPTR;
NODEPTR allocate_node(int item);
int freenode(NODEPTR p);//Deallocates memory space
int push(NODEPTR *top,int item);//pushes an element into the stack
int pop(NODEPTR *top);//pops an element from the stack
//*************************************************************MAIN
FUNCTION**********************************************************
int main()
{
      NODEPTR top=NULL;
      int ch=0,tmp;
      system("clear");
            while(ch!=3)
      {
                  printf("\n\tMenu::\n\t1.Push\n\t2.Pop\n\t3.Exit\n\tYour
choice:: ");
            scanf("%d",&ch);
            switch(ch)
            {
                  case 1: printf("\n\tEnter data item:: ");
                              scanf("%d",&tmp);
                              push(&top,tmp);
                              break;
                  case 2: tmp=pop(&top);
                        if(tmp!=-9999)
                              printf("\n\tThe item popped is:: %d",tmp);
                        else
                              printf("\n\tItem could not be deleted. Stack
empty");
                        break;
                  case 3: printf("\n\tProgram Terminated\n");
                              exit(0);
                              break;
                  default:printf("\n\tIncorrect value entered. Enter
choice again");
            }
            }
            return 0;
}
//*************************************************************MEMBER
FUNCTIONS*********************************************************
NODEPTR allocate_node(int item)
{
      NODEPTR temp = (NODEPTR)malloc(sizeof(struct Node));
      temp->data=item;
      temp->next=NULL;
      return temp;
}
int freenode(NODEPTR p)
{
    free(p);
    p=NULL;
```

```
    return 0;
}
int push(NODEPTR *top,int item)
{
    NODEPTR inter=allocate_node(item);

    /*Displays stack overflow which occurs only if the system couldn't
allocate any more memory from the heap.*/
    if(inter==NULL)
    {
       printf("\n\tMemory couldnt be allocated from the heap. Overflow
occurs.");
       return 1;
    }
    if(*top==NULL)
     {
          *top=inter;
     }
     else
     {
          inter->next=(*top);
          *top=inter;
     }
     return 0;
}
int pop(NODEPTR *top)
{
    NODEPTR p = *top;
    int item;

    /*Returns arbitary value if stack is empty*/
    if(*top==NULL)
    {
     printf("\n\tUnderflow Occurs.");
     return -9999;
    }
    else
    {
     *top=(*top)->next;
     item=p->data;
     freenode(p);
     return item;
    }
}
```

OUTPUT:

```
     Menu::
     1.Push
     2.Pop
     3.Exit
     Your choice:: 1

     Enter data item:: 10
```

```
Menu::
1.Push
2.Pop
3.Exit
Your choice:: 1

Enter data item:: 20

Menu::
1.Push
2.Pop
3.Exit
Your choice:: 2

The item popped is:: 20
Menu::
1.Push
2.Pop
3.Exit
Your choice:: 2

The item popped is:: 10
Menu::
1.Push
2.Pop
3.Exit
Your choice:: 2

Underflow Occurs.
Item could not be deleted. Stack empty
Menu::
1.Push
2.Pop
3.Exit
Your choice:: 3

Program Terminated
```

DISCUSSION:

1.The complexity of push is O(1).
2.The complexity of pop is O(1).
3.Overflow occurs only when the program cannot be allocated any more memory by the system.