Stack And Queue

DATE 21/2/14

Stack : It is defined as a list of elements in which we can insert or delete the element only at the top of the stack.

eg. in our daily life, we use a stack of CD's, stack of books. and we can take CD or book only from the top.
Similarly we can keep one more CD or book only at the top.

There are two operations possible on the stack.

① Push, when we add the item in stack.
② Pop, when we delete the item from stack.

The elements which are added (pushed) at the last in stack can be deleted (popped) from the stack, so its behaviour is like last in first out. So, stack is also called LIFO data structu

We can implement stack in two ways -

(i) Array implementation of stack.
(ii) Linked list implementation of stack.

Array implementation of stack :

In array, any element can be added or deleted at any place and we want to push or pop the element from top of the stack only. So we take a variable top, which keeps the position of the top element in array.
In stack, we have to check two possibility cases.

1) when there is no place for adding (push) the element in the array, then this condition is called overflow.
So, we have to check the value of top with size of arr

2) when there is no element for deleting (pop) from stack, then this condition is called underflow. If there is no element in stack then value of top will be -1. So, we have to che the value of top before deleting the element of stack

Push Operation On stack :

```
if (top == (MAX-1))
        printf ("stack overflow\n");
else
{
        top = top + 1;
        stack_arr [top] = pushed_item;
}
```

Pop operation On stack :

```
if (top == -1)
    printf ("stack Underflow\n");
else
{   printf ("Popped element is: %d\n", stack_arr[top]);
    top = top - 1;
}
```

ii) Linked list Implementation of Stack :

Stack can be implemented through linked list.
For this we will take the structure as -

```
struct node {   int info;
                struct node *link;
            }
```

Push Operation on stack:

For pushing the element on stack we follow the insertion operation of the linked list, means we add the element at the start of the list.

```
temp -> info = pushed_item;
temp -> link = top;
top = temp;
```

Here top always points to the first node of linked list. After statement 3, last pushed item will become first node of linked list. Since stack implementation is through linked list so we don't check overflow condition.

Pop operation on Stack:

For pop operation on stack we delete the first element of linked list because stack is a last in first out structure.

```
if (top == NULL)
    printf ("stack is empty\n");
else
    {
        temp = top;
        printf ("Popped item is %d\n", temp->info);
        top = top -> link;
        free (temp);
    }
```

Here first we check for stack empty condition then we pop the first element of stack.
Now top will point to second node of linked list which will become first node of linked list.

Q6 Write a program of stack using array.

```
#include <stdio.h>
#include <conio.h>
#define MAX 5
int top = -1;
int stack [MAX];
int main()
{
    char ch;
    int choice;
    do
    {
        printf ("1. Push\n");
        printf ("2. Pop\n");
        printf ("3. Display\n");
        printf ("4. Quit\n");
        printf ("Enter your choice:");
        scanf ("%d", &choice);
        switch (choice)
        {
            case 1: push();
                    display();
                    break;
            case 2: pop();
                    display();
                    break;
            case 3: display();
                    break;
            case 4: exit();
            default: printf ("wrong choice\n");
        }
        printf ("\n Do you want to continue (Y/N):");
        ch = getch();
    }
    while (ch == 'y');
    getch();
```

```
void push (int pushed_item)
{
    if (top == (MAX-1))
        printf ("stack overflow \n");
    else
    {
        printf ("Enter the item to be pushed in stack:");
        scanf ("%d", &pushed_item);
        top = top + 1;
        stack [top] = pushed_item;
    }
}
int pop ()
{
    if (top == -1)
        printf ("stack underflow \n");
    else
    {
        printf ("Popped element is: %d \n", stack[top]);
        top = top - 1;
    }
}
display ()
{
    int i;
    if (top == -1)
        printf ("stack is empty \n");
    else
    {
        printf ("stack elements are: \n");
        for (i = top; i >= 0; i--)
            printf ("%d \n", stack_arr[i]);
    }
}
```

Q 7. Write a program of stack using linked list :

```
#include <stdio.h>
#include <malloc.h>
#include <conio.h>
  struct node
  {   int info;
      struct node * link;
  } *top = NULL;
int main()
  {  char ch;
  {  int choice ;
     do
      {  printf ("1. Push \n");
         printf ("2. Pop\n");
         printf ("3. Display\n");
         printf ("4. Quit \n");
         printf ("Enter your choice : ");
         scanf ("%d", &choice);
         switch (choice)
         {
            case 1: push();
                    break;
            case 2: pop();
                    break;
            case 3: display();
                    break;
            case 4: exit()
            default: printf ("wrong choice\n");
         }
         printf ("\n Do you want to continue (Y/N) : ");
         ch = getche();
      }
     while (ch == 'y');
  } getch();
```

DATE

```
void push (int pushed_item)
  {
      struct node *temp;
      temp = (struct node *) malloc (sizeof (struct node));
      printf (" Input the new value to be pushed on the
              stack :");
      scanf ("%d ", & pushed_item);
      temp -> info = pushed_item ;
      temp -> link = top;
      top = temp;
  }
pop ()
{
    struct node *temp;
    if (top == NULL)
        printf (" stack is empty \n");
    else
      {   temp = top;
          printf (" popped item is %d \n", temp -> info);
          top = top -> link;
          free (temp);
      }
}
display ()
  {   struct node *ptr;
      ptr = top;
      if (top == NULL)
          printf ("stack is empty \n");
      else
        {   printf ("stack elements: \n");
            while (ptr != NULL)
```