Stack

A stack can be fixed in size or have a dynamic implementation in which the size can change. Attempting to add an element to an already full stack in the case of bounded capacity stacks results in a stack overflow exception. Stack overflow is essential for the GATE exam as well. Underflow is a condition that occurs when a pop operation attempts to remove an element from an already empty stack.

Stack Definition

A stack is an ordered collection of items into which new items may be inserted and from which items may be deleted at one end, called the TOP of the stack. Stack is a LIFO (Last In, First Out) data structure.

Operations on Stack

Stack operations may include initializing the stack, using it, and then de-initializing it. Aside from these fundamental operations, a stack is used for the two primary operations listed below, which are discretley elaborated in the GATE CSE syllabus as well:

- **Push**
- **Pop**

To use a stack efficiently, we must also check the stack's status. Stacks have the following functionality added for the same purpose.

- **size()**
- **isEmpty()**
- **isFull**

The operations work as follows:

1. A pointer called TOP is used to keep track of the top element in the stack.
2. When initializing the stack, we set its value to -1 to check if the stack is empty by comparing TOP == -1.
3. On pushing an element, we increase the value of TOP and place the new element in the position pointed to by TOP.
4. On popping an element, we return the element pointed to by TOP and reduce its value.
5. Before pushing, we check if the stack is already full.
6. Before popping, we check if the stack is already empty.

Implementation of Stack

A stack can be implemented using two ways: Array and Linked list. Since array size is defined at compile time, it can't grow dynamically. Therefore, an attempt to insert/push an element into the stack (implemented through an array) can cause a stack overflow situation if it is already full.

But if the stack is implemented using the linked list to avoid the stack overflow problem, we need to use a linked list to implement a stack because the linked list can grow dynamically and shrink at runtime.

Push and Pop Implementation Using Array

The PUSH operation involves inserting a new element into a Stack. Because a new element is always inserted from the topmost position of the Stack, we must always check to see if the top is empty, i.e., TOP = Max-1. If this condition is false, the Stack is full, and no more elements can be inserted; if we try to insert the element, a Stack overflow message is displayed.

Array implementation aims to create an array where the first element inserted is placed at stack[0] and will be deleted last. In array implementation, the element inserted at the top must be kept.

The algorithm for the push is as follows:

```
void push( )

{

if(top==max)

printf("\nOverflow");

else

{

int element;

printf("\nEnter Element:");

scanf("%d",&element);

printf("\nElement(%d) has been pushed at %d", element, top);

stack[++top]=element;

}

}
```

POP denotes the removal of an element from the Stack. Before deleting an element, ensure that the Stack Top, i.e., TOP=NULL, is not NULL. If this condition is met, the Stack is empty, and no deletion operations can be performed; if we try to delete, the Stack underflow message is generated.

The algorithm for the pop is as follows:

```
void pop( )

{

if(top==-1)

printf("\nUnderflow");

else

{

top--;

printf("\nElement has been popped out!");

}

}
```

Push and Pop Implementation Using Linked List

It is also called dynamic implementation, as the stack size can grow and shrink as the elements are added or removed respectively from the stack.

- The PUSH operation on the stack is the same as inserting a node in a Single linked list.
- The POP operation is the same as deleting a node in a Single linked List.

```
struct node
{

int data;

struct node *prev;

}

*top=NULL, *temp=NULL;

void push( )

{

temp = (struct node*)malloc(sizeof(struct node*));

printf("\nEnter Data:");

scanf("%d",&temp->data);

temp->prev=NULL;

if(top==NULL)

{
```

```
top=temp;

}

else

{

temp->prev=top;

top=temp;

}

}

void pop( )

{

temp=top->prev;

top=temp;

printf("\nDeleted: %d",top->prev);

}
```

Applications of Stack

The application of the data structure stack is used for backtracking and parenthesis matching. The stack is also used to implement the Depth-first Search. Syntax analysis of the compiler uses stack in parsing the program.

Web browsers store the addresses of recently visited sites on a stack. The stack can be used for checking the syntax of an expression as:

- **Infix expression:** It is the one where the binary operator comes between the operands. e. g., A + B * C.
- **Postfix expression:** Here, the binary operator comes after the operands. e.g., ABC * +
- **Prefix expression:** Here, the binary operator proceeds with the operands. e.g.,+ A * BC. This prefix expression is equivalent to A + (B * C) infix expression. Prefix notation is also known as Polish notation. Postfix notation is also known as a suffix or Reverse Polish notation.

Advantages of Stack

Stack supports the management of data that uses the LIFO method. Stacks are used to manage memory systematically. It is used in many virtual machines, such as the JVM. When a function is invoked, the local variables and other function parameters are stored in the stack and are automatically destroyed when the function returns.

Stacks are more secure and reliable because they do not get easily corrupted. The stack provides control over memory allocation and deallocation. Stack automatically cleans up the objects.

Disadvantages of Stack

The disadvantage of the stack is that the stack memory size is limited. The total size of the stack must be defined ahead of time of the declaration if it is implemented using the array data structure.

The stack overflow can occur in the program if too many objects are created in the class. In a stack, random access is not possible, as we can access only the top of the stack. If the stack falls outside of memory, this can result in an abnormal termination.