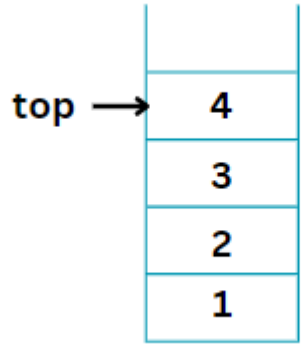
Module -2
Stack: Data Structure

Introduction

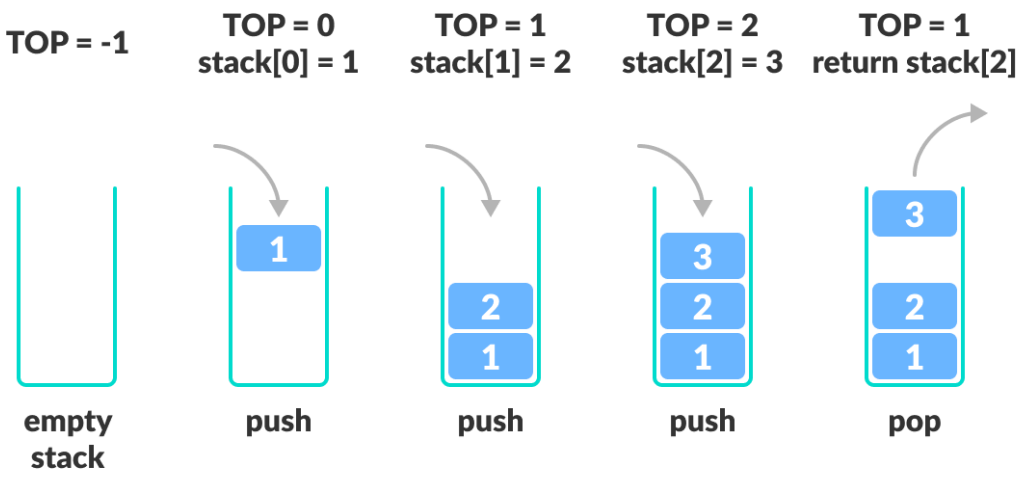
- Stack is a linear data structure that follows either LIFO (Last In First Out) or FILO (First In Last Out) principle.
- This means that the last element that is inserted is the first element to be removed.
- The stack is used to solve a few of the general problems like:
  1. Tower of Hanoi
  2. N-Queens Problem
  3. Infix to Prefix Conversion



Operations Performed on Stacks

The following are the basic operations served by stacks.

- push: Adds an element to the top of the stack.
- pop: Removes the topmost element from the stack.
- isEmpty: Checks whether the stack is empty.
- isFull: Checks whether the stack is full.
- peek: Displays the topmost element of the stack.



```
int MAXSIZE = 8;
int stack[8];
int top = -1;
```

Operations Performed on Stacks

isEmpty: Checks whether the stack is empty

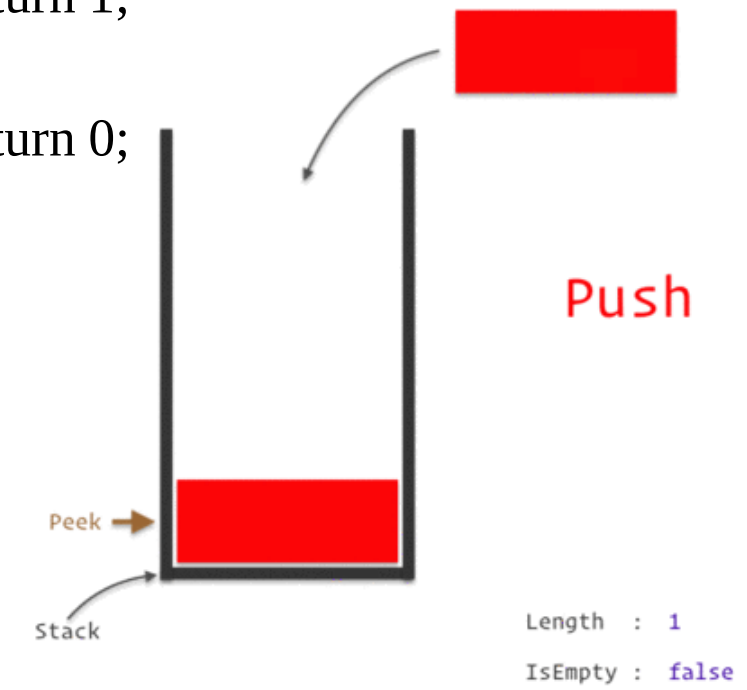
```
/* Check if the stack is empty */
int isempty(){
   if(top == -1)
      return 1;
   else
      return 0;
}
```

isFull: Checks whether the stack is full.

```
/* Check if the stack is full*/
int isfull(){
   if(top == MAXSIZE)
      return 1;
   else
      return 0;
}
```

peek: Displays the topmost element of the stack.

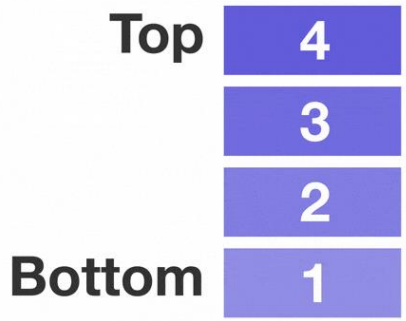
```
int peek(){
   return stack[top];
}
```



Operations Performed on Stacks

push: Adds an element to the top of the stack.

```
/* Function to insert into the stack */
int push(int data){
   if(!isfull()) {
      top = top + 1;
      stack[top] = data;
   } else {
      printf("Could not insert data,
Stack is full.\n");
   }
}
```

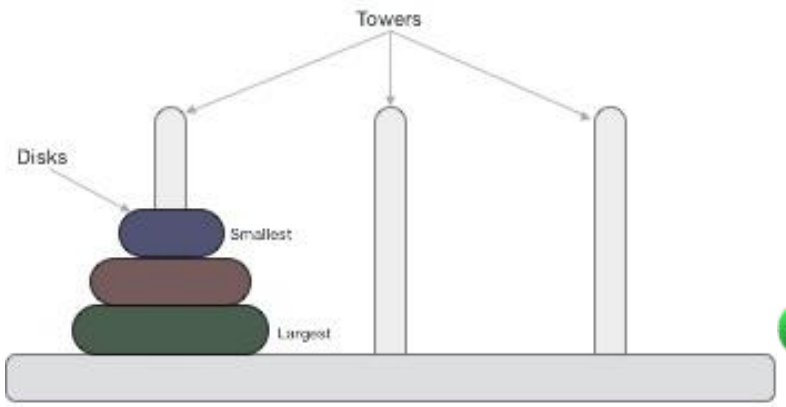


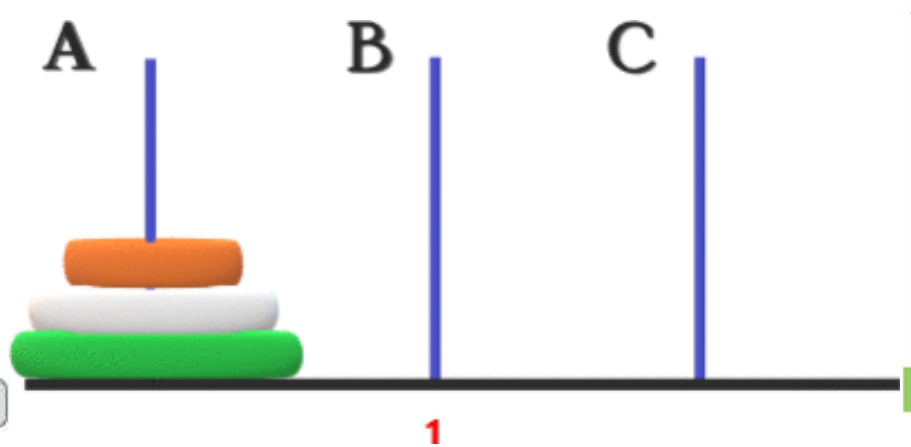
pop: Removes the topmost element from the stack.

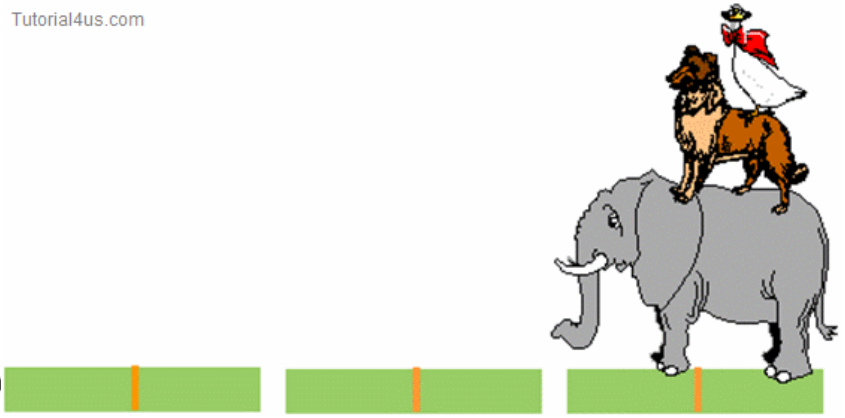
```
/* Function to delete from the stack */
int pop(){
   int data;
   if(!isempty()) {
      data = stack[top];
      top = top - 1;
      return data;
   } else {
      printf("Could not retrieve data, Stack is
empty.\n");
   }
}
```

Problem: Tower of Hanoi

- The Tower of Hanoi is a mathematical puzzle. It consists of three rods and N disks. The task is to move all disks to another rod following certain rules:
    1. Only one disk can be moved at a time.
    2. Only the uppermost disk can be moved from one stack to the top of another stack or to an empty rod.
    3. Larger disks cannot be placed on top of smaller disks.







Problem: Tower of Hanoi

```
void towerOfHanoi(int n, char from_rod, char to_rod, char aux_rod){
    if (n == 1){
        cout << "Move disk 1 from rod " << from_rod <<
                        " to rod " << to_rod<<endl;
        return;
    }
    towerOfHanoi(n - 1, from_rod, aux_rod, to_rod);
    cout << "Move disk " << n << " from rod " << from_rod <<
                            " to rod " << to_rod << endl;
    towerOfHanoi(n - 1, aux_rod, to_rod, from_rod);
}
solve(){
     TowerOfHanoi(n, 'A', 'C', 'B');
}
```

Problem: Tower of Hanoi

$$T(n) = 2\,T(n-1) + 1$$

$$T(n) = 2 \qquad\qquad + 1$$

$$T(n) = 2\,[\,2\,T(n-2) + 1\,] + 1$$

$$T(n) = 2\,[\,2 \qquad\quad + 1\,] + 1$$

$$T(n) = 2\,[\,2\,[\,2\,T(n-3) + 1\,] + 1\,] + 1$$

$$T(n) = 2\,[\,2\,[\,2\,[\,2\,T(n-4) + 1\,] + 1\,] + 1\,] + 1$$

$$T(n) = 2^4\,T(n-4) + 15$$

$$\cdots$$

$$T(n) = 2^k\,T(n-k) + 2^k - 1$$

Since n is finite, k -> n. Therefore,

$$\lim_{k \to n} T(n) = 2^n - 1$$