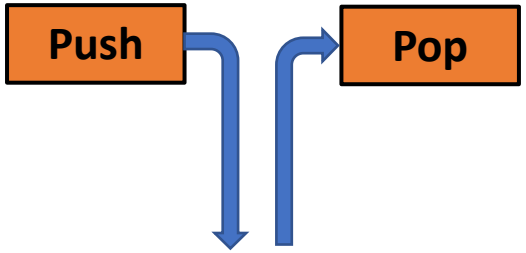
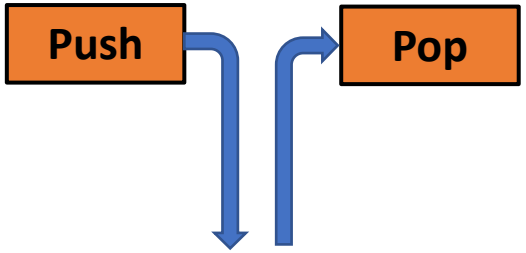
# Outline : Lecture 6

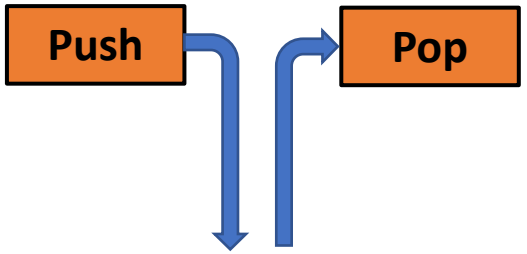
**Stack (LIFO)**

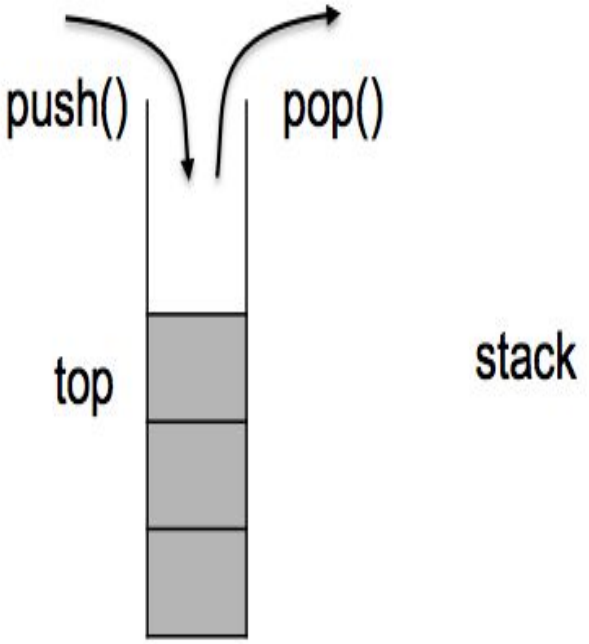
- Static Stack & Dynamic Stack
- Implementation of Static Stack
  - ✔ Push
  - ✔ Pop
  - ✔ Display
- Implementation of Dynamic Stack
  - ✔ Push
  - ✔ Pop
  - ✔ Display
- Application of stack
  - ✔ Transformation of Infix Arithmetic Expression into Equivalent Postfix Expression
  - ✔ Evaluation of Postfix Expression
- Recursion
  - ✔ Tower of Hanoi Problem

# Stack (LIFO):



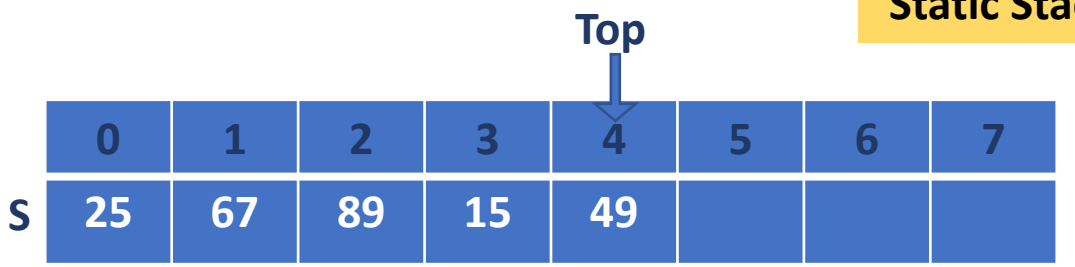



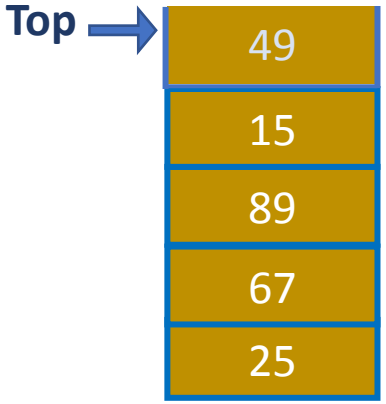



# Static Stack & Dynamic Stack

**Static Stack**

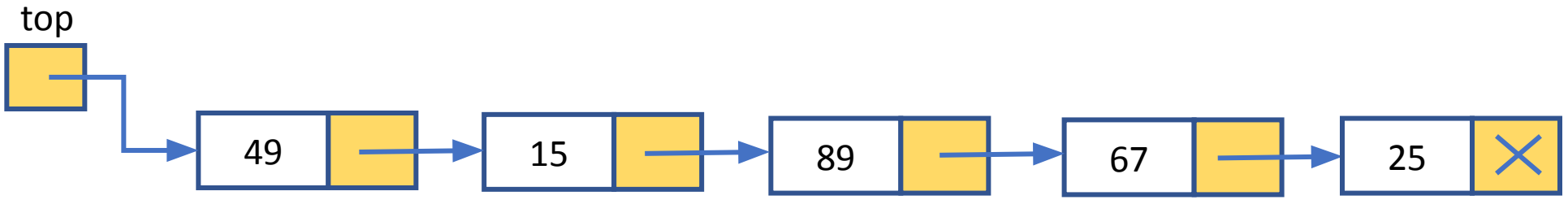



```
#define MAX 8;
typedef struct stack
 {
    int top;
    int s[MAX];
  }stk;
```



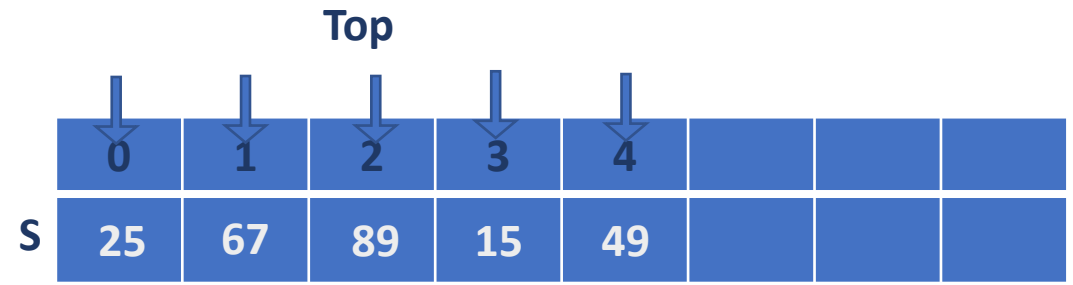

```
typedef struct stack_node
 {
    int data;
    struct stack_node *link;
  }stk;

stk *top=NULL;
```

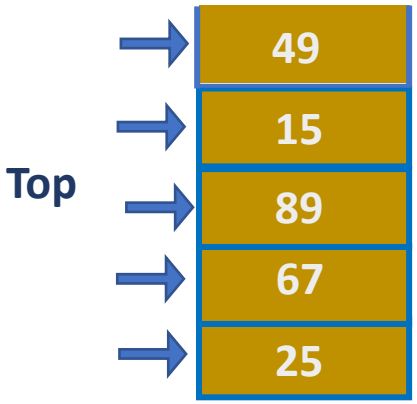
**Dynamic Stack**



# Implementation of Static Stack (using Array):

```
#define MAX 5;
typedef struct stack
 {
    int top;
    int s[MAX];
  }stk;
```



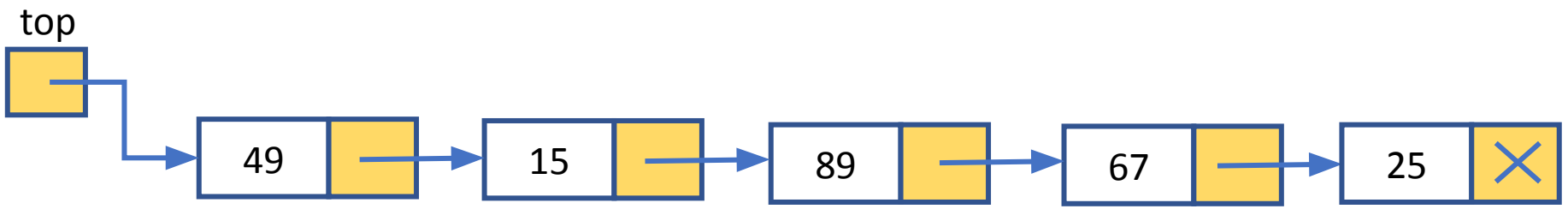




```
void push(int x)
    {
    if(top == MAX − 1)
     printf("\nOverflow");
   else
     {
     top = top + 1;
     s[top] = x;
     }
    }
```

```
int pop(stk stack1)
{
      if(stack1.top == −1)
       printf("\n Underflow");
   else
{
     x = stack1.s[stack1.top];
  stack1.top = stack1top − 1;
  return(x);
    }
}
```

```
void push(int x, stk stack1)
{
      if(stack1.top == MAX − 1)
       printf("\nOverflow");
    else
  {
  stack1.top = stack1top + 1;
  stack1.s[stack1.top] = x;
   }
}
```

# Dynamic Stack




```
typedef struct stack_node
 {
    int data;
    struct stack_node *link;
  }stk;

stk *top=NULL;
```

```
void push(int x)
      {
        stk  * ptr;
        ptr = (stk *)mallock(sizeof(stk));
        ptr → data = x;
        ptr → link = top;
            top = ptr;
      }
```

```
int pop()
      {
         stk *ptr;
         if(top)
          {
            ptr=top;
            top=top→link;
            free(ptr); // deallocating memory
           }
        else
            printf("/n Underflow");
       }
```