K.R. MANGALAM UNIVERSITY

THE COMPLETE WORLD OF EDUCATION

Data Structure

ENCS205

School of Engineering & Technology (SOET)
K.R. MANGALAM University

UNIT-2

Session 23: Implementing Stacks (Array & Linked List)

Session 23 Outlook

- Background
- Basic Introduction of stack
- Operations of stack using arrays
- Practice Question
- Operations of stack using linked list

Key Learning Outcomes

- Students should be able to recall stack terminology and ADT operations.
- Apply array-based and linked list based stack operations and conversion algorithms.
- Students should evaluate efficiency of stack operations and algorithms. Evaluate impact of stack structure on complexity.

Array representation of Stack

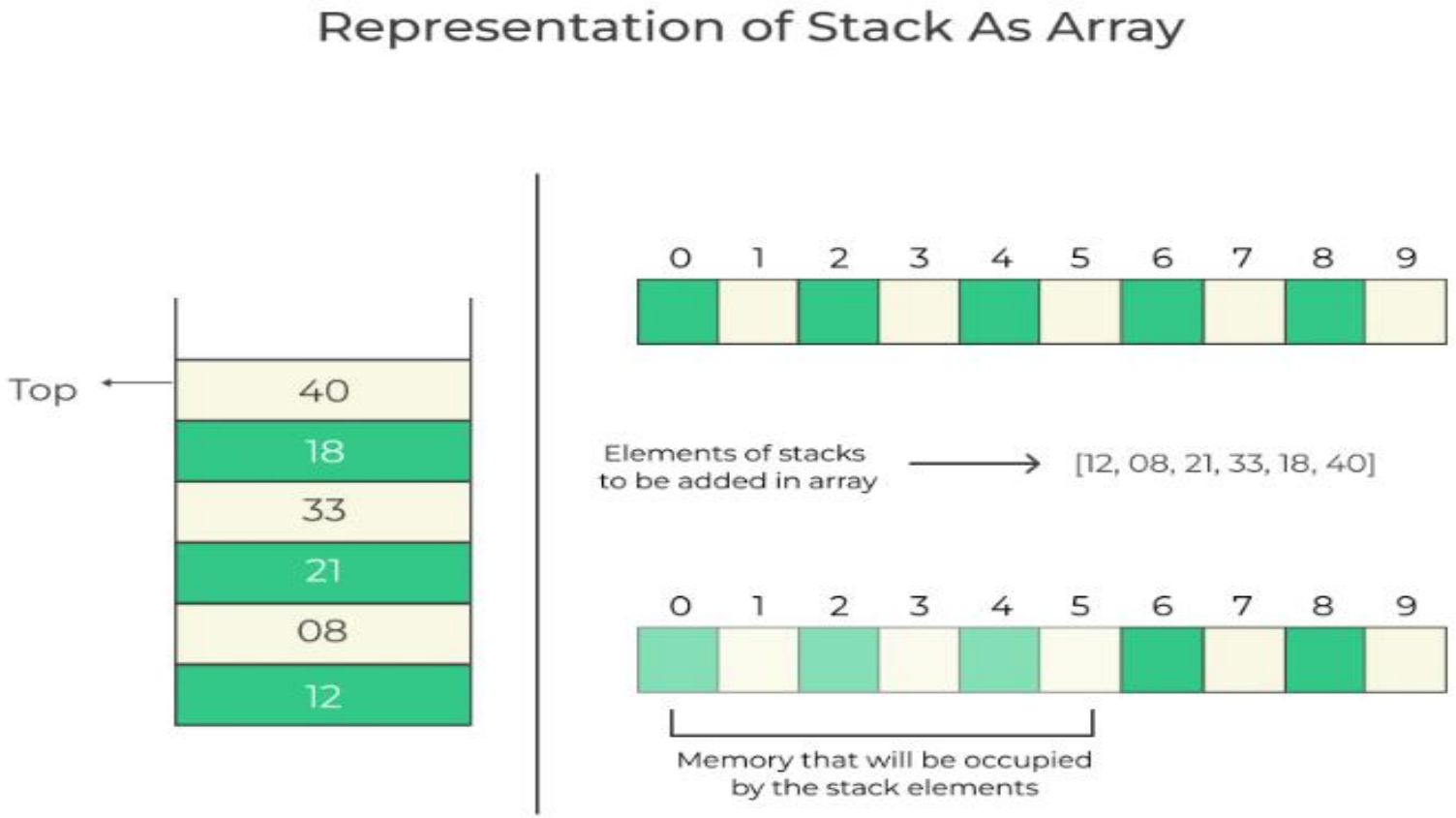
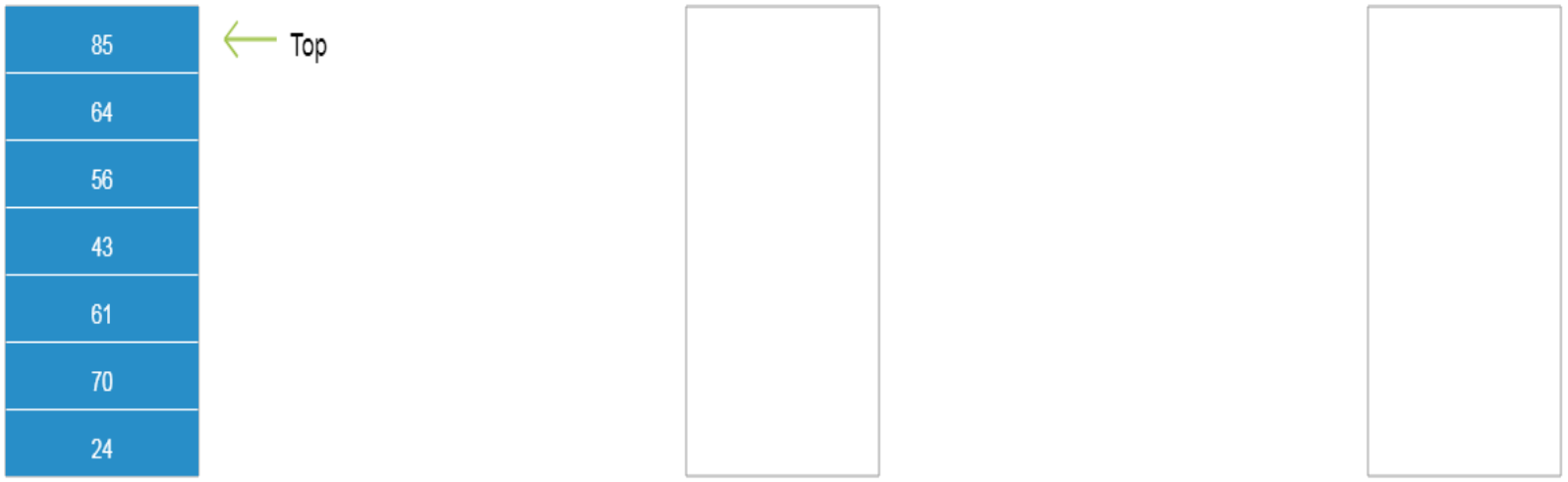


Fig: Array Representations

https://prepinsta.com/data-structures/representation-of-a-stack/

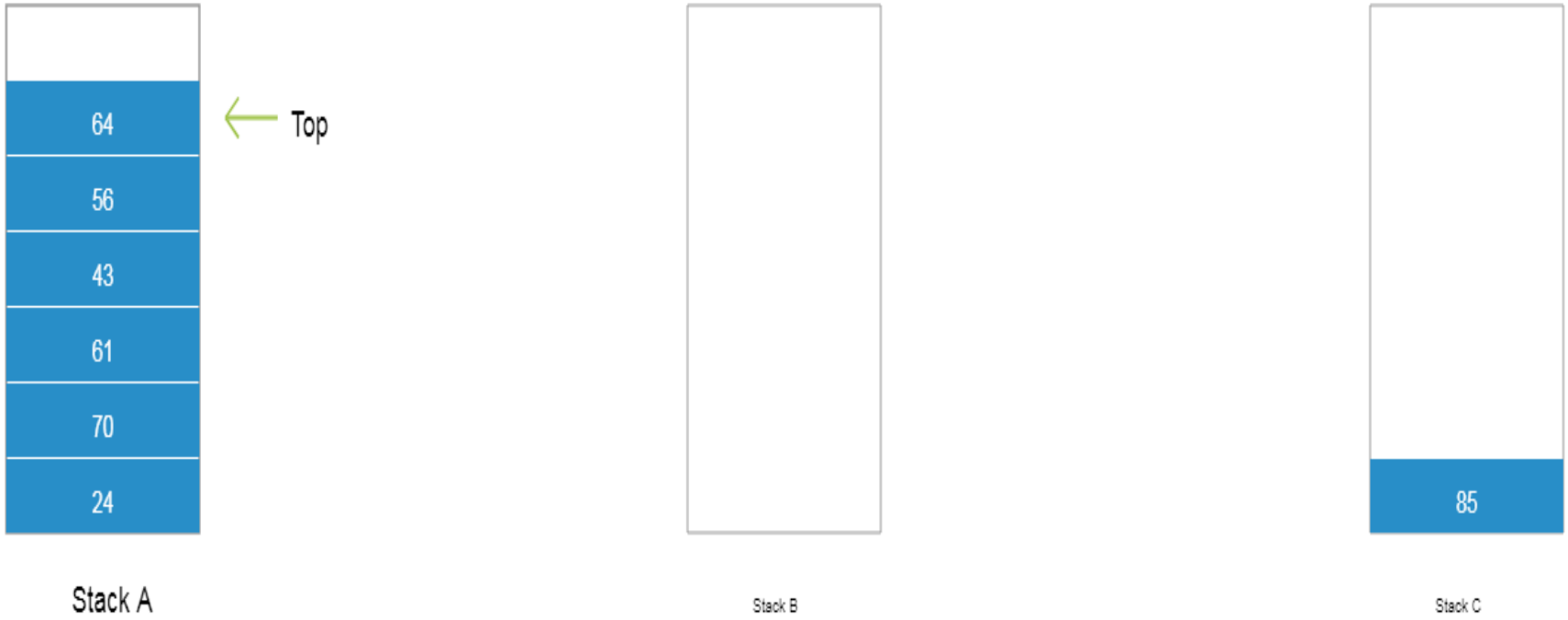
Working of stack

Observe: Stack A with random numbers is given. Observe how the prime numbers from stack A are popped and pushed into stack B whereas the others are pushed into stack C.



Observations

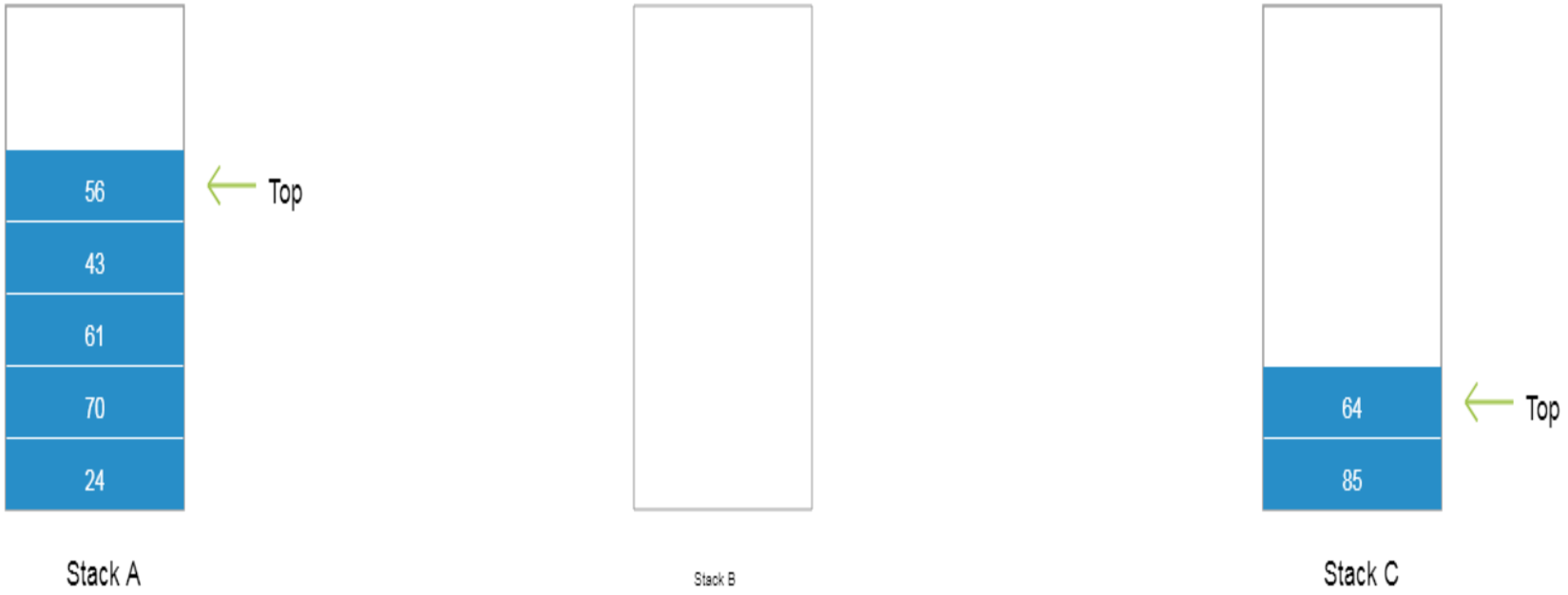
Working of stack



Observations

The number popped from stack A (i.e 85) is not a prime, so it is pushed into stack C

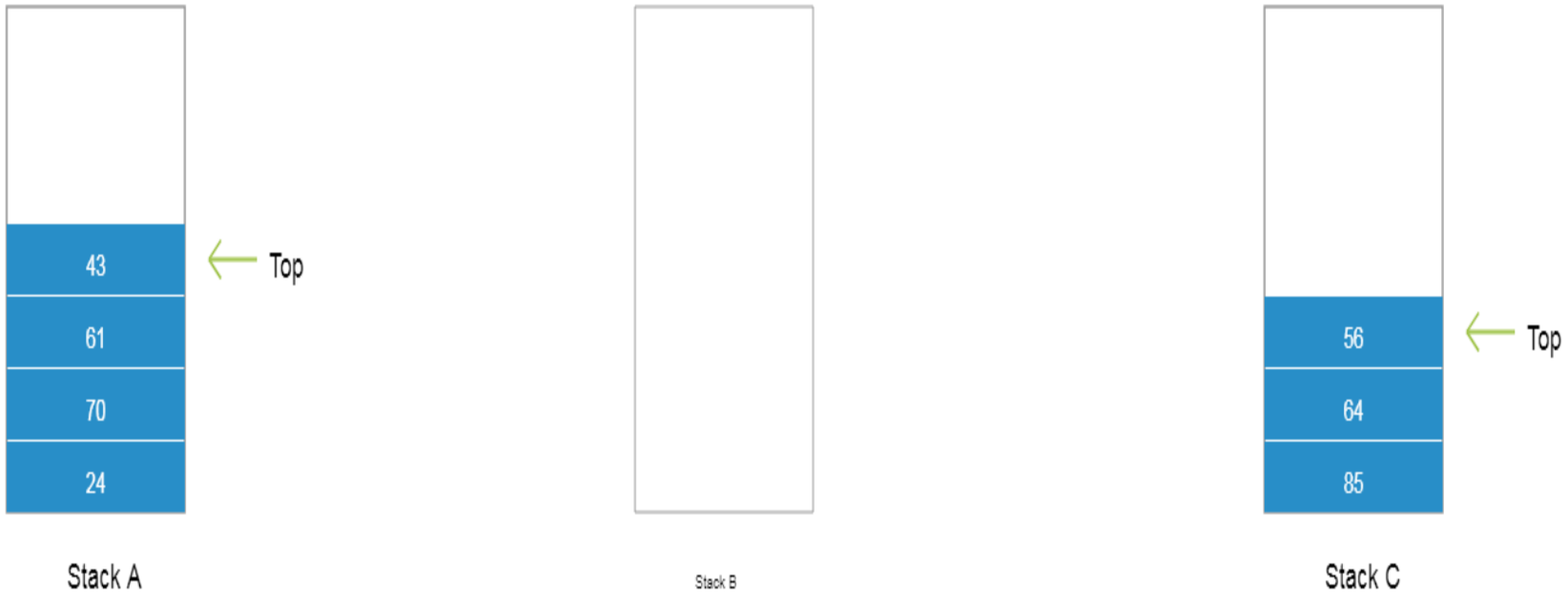
Working of stack



Observations

The number popped from stack A (i.e 64) is not a prime, so it is pushed into stack C

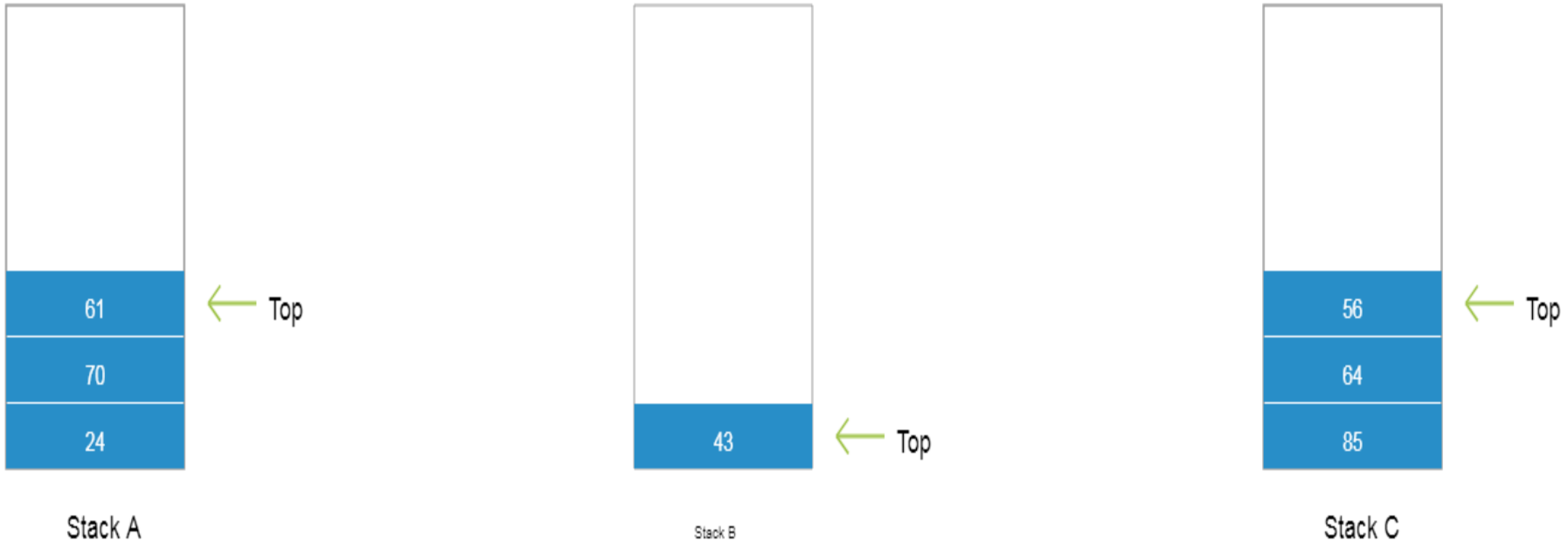
Working of stack



Observations

The number popped from stack A (i.e 56) is not a prime, so it is pushed into stack C

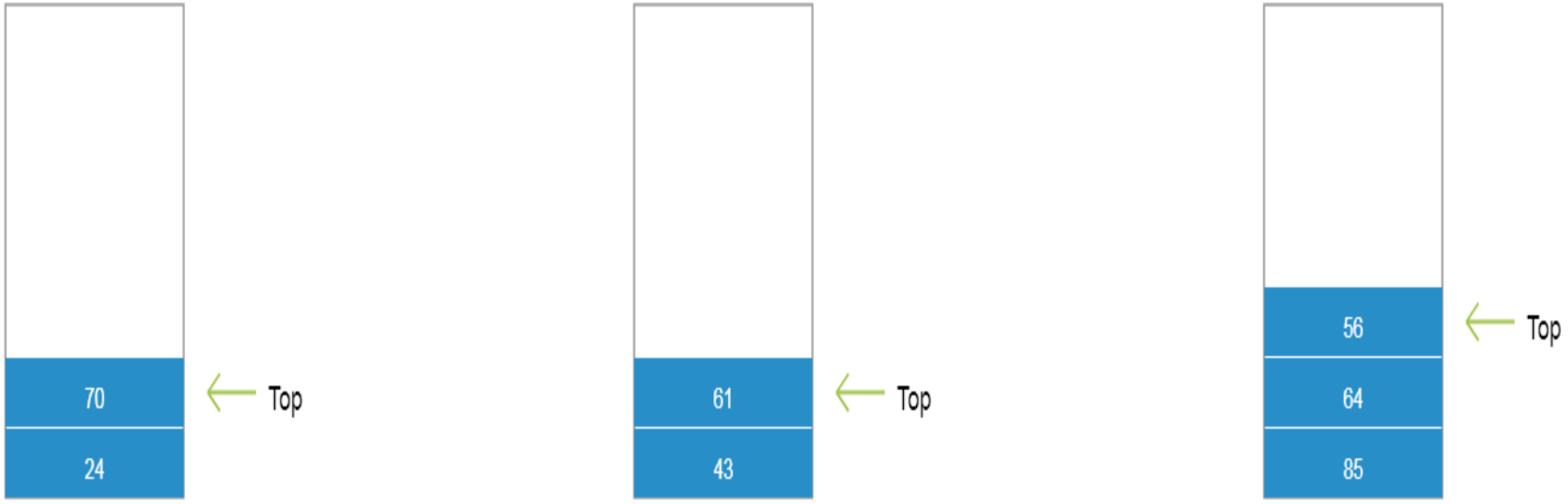
Working of stack



Observations

The number popped from stack A (i.e 43) is prime, so it is pushed into stack B

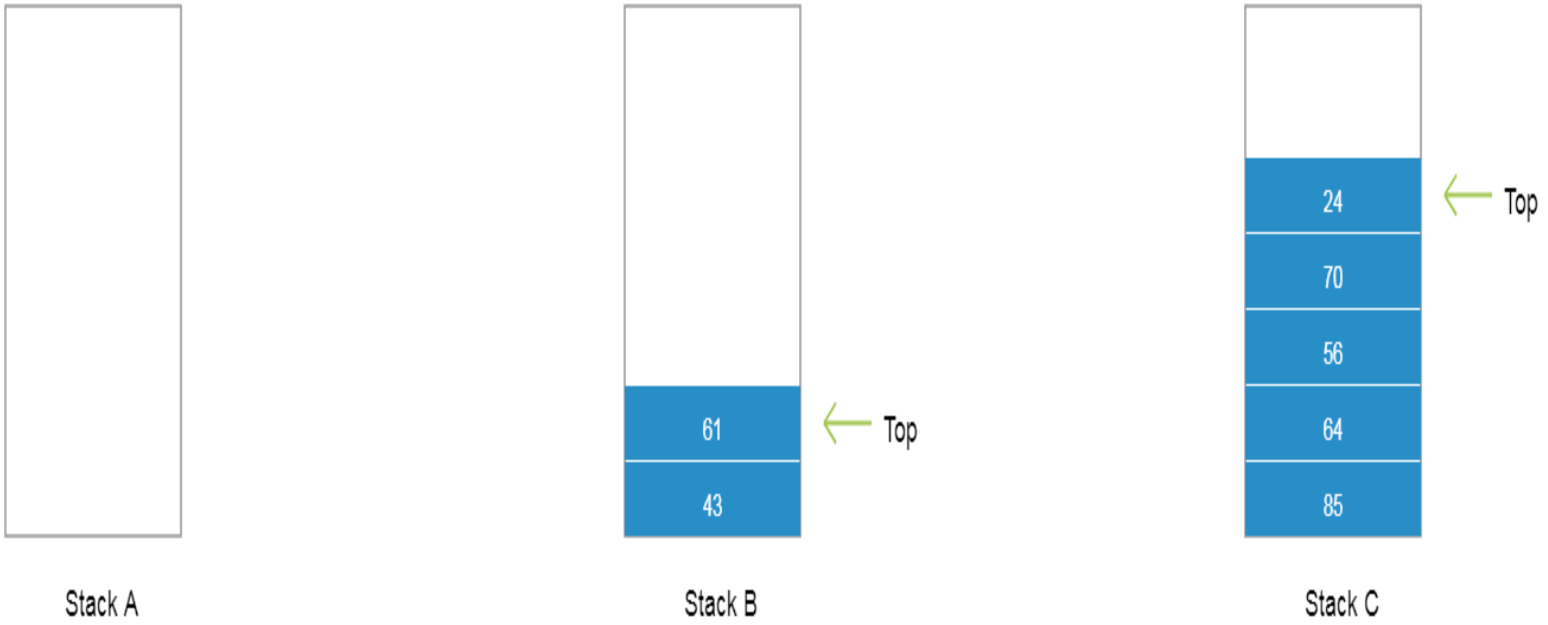
Working of stack



Observations

The number popped from stack A (i.e 61) is prime, so it is pushed into stack B

Working of stack



Stack A

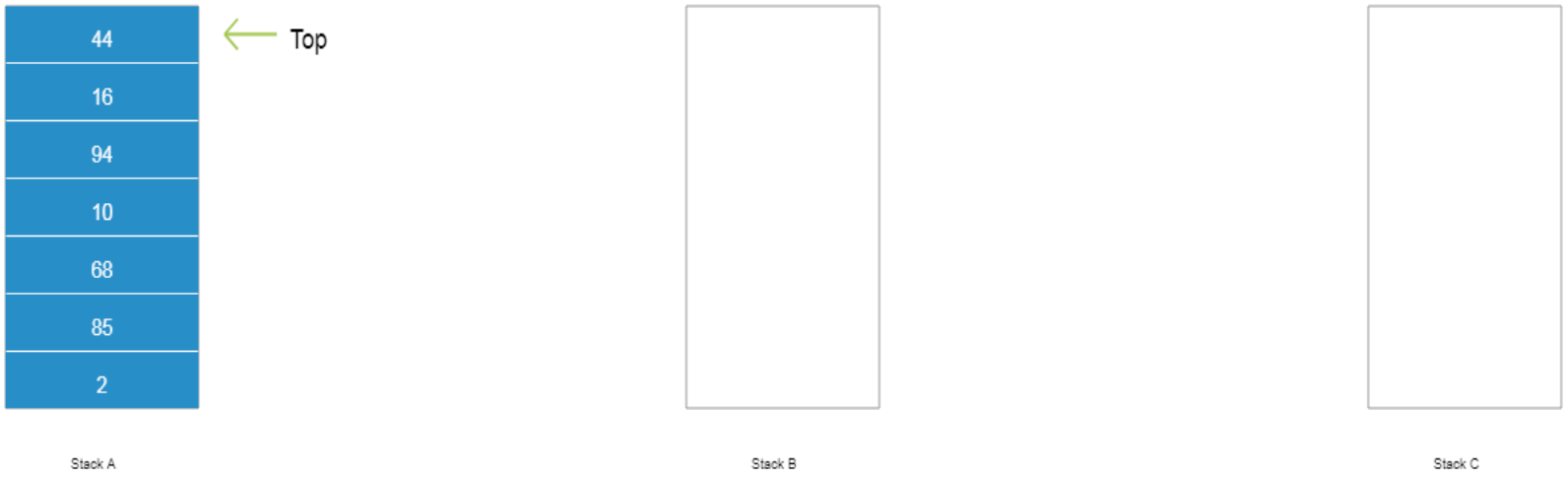
Stack B

Stack C

Observations

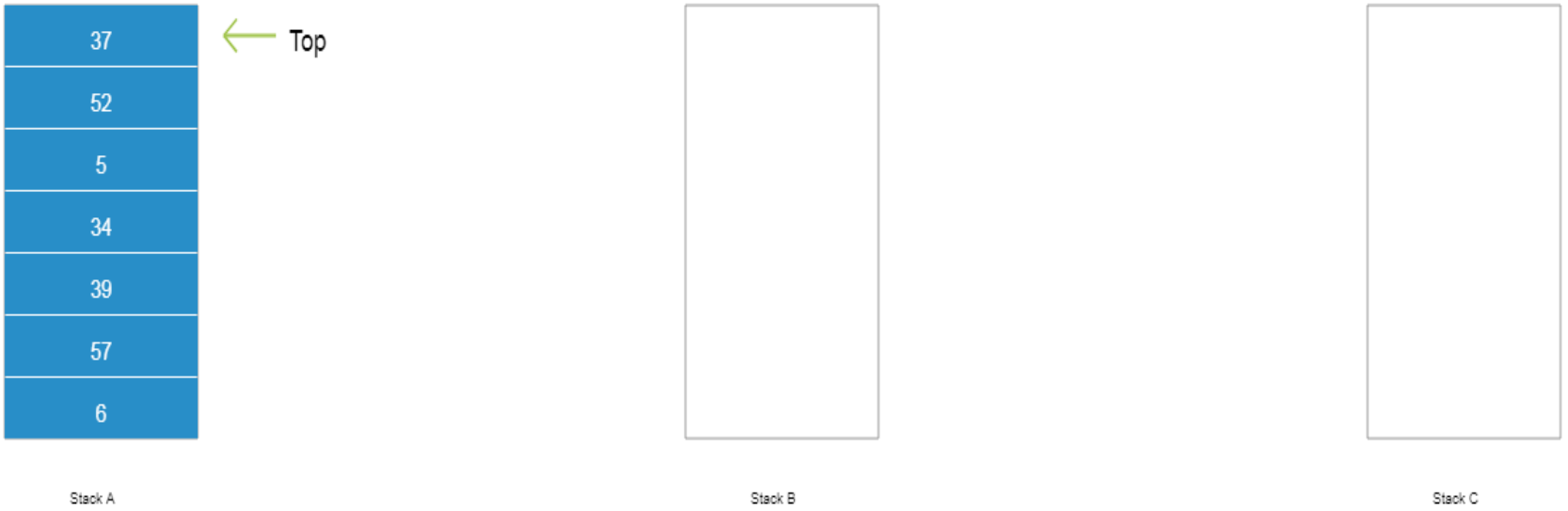
As stack A is empty.
Demonstration is complete!!.

Observe: Stack A with random numbers is given. Observe how the prime numbers from stack A are popped and pushed into stack B whereas the others are pushed into stack C.



Observations

Observe: Stack A with random numbers is given. Observe how the prime numbers from stack A are popped and pushed into stack B whereas the others are pushed into stack C.



Observations

Why Implement Stack Using Array?

- Simple and Direct Access
  - **Indexing**
  - **Time Complexity**
- Predictable Memory Usage
  - **Fixed Size**
  - **Memory Allocation**
- Ease of Implementation
  - **Simplicity**:
  - **Less Overhead**
- Performance
  - **Efficient Operations**
  - **No Pointer Management**

Limitations of Stack Implementation Using Array

- Fixed Size
- Wasted Space
- Resizing Issues

Code

```
#include <iostream.h>
using namespace std;
int stack[100], n=100, top=-1;
void push(int val) {
  if(top>=n-1)
  cout<<"Stack Overflow"<<endl;
  else {
    top++;
    stack[top]=val;
  }
}
```

```
void pop() {  if(top<=-
1)  cout<<"Stack
Underflow"<<endl;  else
{    cout<<"The popped
element is "<< stack[top]
<<endl;    top--;  }
}
```

Code

```
void display() {
  if(top>=0) {
    cout<<"Stack elements are:";
    for(int i=top; i>=0; i--)
    cout<<stack[i]<<" ";
    cout<<endl;
  } else
  cout<<"Stack is empty";
}
```

```
int main() {
  int ch, val;
  cout<<"1) Push in
stack"<<endl;
  cout<<"2) Pop from
stack"<<endl;
  cout<<"3) Display
stack"<<endl;
  cout<<"4) Exit"<<endl;
  do {
    cout<<"Enter choice:
"<<endl;
    cin>>ch;
```

Conti..

```
switch(ch) {
        case 1: {
          cout<<"Enter value to be
pushed:"<<endl;
          cin>>val;
          push(val);
          break;
        }
        case 2: {
          pop();
          break;
        }
        case 3: {
          display();
          break;
        }
case 4: {
          cout<<"Exit"<<endl;
          break;
        }
 default: {          cout<<"Invalid
Choice"<<endl;
 }     }  }
while(ch!=4);    return 0;
 }
```

Output

1) Push in stack
2) Pop from stack
3) Display stack
4) Exit
Enter choice: 1
Enter value to be pushed: 2
Enter choice: 1
Enter value to be pushed: 6
Enter choice: 1
Enter value to be pushed: 8
Enter choice: 1
Enter value to be pushed: 7
Enter choice: 2
The popped element is 7
Enter choice: 3
Stack elements are:8 6 2
Enter choice: 5
Invalid Choice
Enter choice: 4
Exit

Test Your self

1. Let S be a stack of size n >= 1. Starting with the empty stack, suppose we push the first n natural numbers in sequence, and then perform n pop operations. Assume that Push and Pop operation take X seconds each, and Y seconds elapse between the end of one such stack operation and the start of the next operation.

For m >= 1, define the stack-life of m as the time elapsed from the end of Push(m) to the start of the pop operation that removes m from S.

The average stack-life of an element of this stack is

a) n(X+ Y)
b) 3Y + 2X
c) n(X + Y)-X
d) Y + 2X

Test Your self

Solution:

Let's n = 3

1 (Pushed in Stack) : Time take = X

2 (Pushed in Stack) : Time take = X

3 (Pushed in Stack) : Time take = X

3 (Popped Out of Stack) : Time take = X

2 (Popped Out of Stack) : Time take = X

1 (Popped Out of Stack) : Time take = X

Stack Life of 3 = y

Stack Life of 2 = 3(x + y) - x

Stack Life of 1 = 5(x + y) - x

Average Stack Life of Element = (Y + (3(X + Y) - X) + (5(X + Y) - X)) /3 = (6X + 9Y)/3 = 2X + 3Y

For n= 3 the average stack life is 2X +3Y

Similarly for n elements, the average stack life will be **n(X + Y) -X**

2. A single array A[1..MAXSIZE] is used to implement two stacks. The two stacks grow from opposite ends of the array. Variables top1 and top2 (topl< top 2) point to the location of the topmost element in each of the stacks. If the space is to be used efficiently, the condition for "stack full" is:

a) (top1 = MAXSIZE/2) and (top2 = MAXSIZE/2+1)
b) top1 + top2 + 1 = MAXSIZE
c) (top1= MAXSIZE/2) or (top2 = MAXSIZE)
d) top1= top2 -1

3. Which of the following permutation can be obtained in the same order using a stack assuming that input is the sequence 5, 6, 7, 8, 9 in that order?

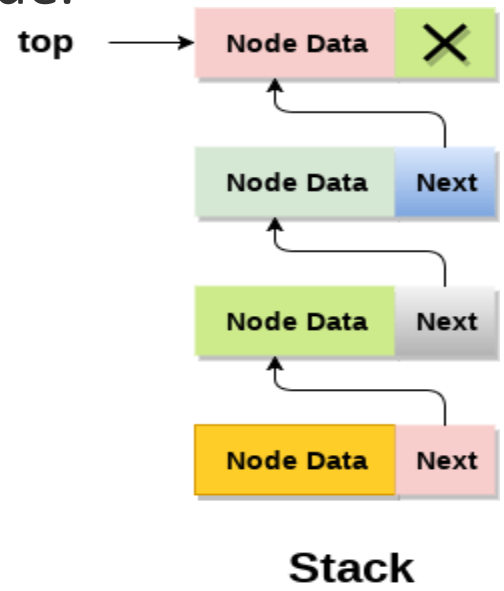
a) 7, 8, 9, 5, 6
b) 5, 9, 6, 7, 8
c) 7, 8, 9, 6, 5
d) 9, 8, 7, 5, 6

Answers:

1. Correct Option: C
2. Correct Option : d
3. Correct Option : C

Implementation of Stack using Linked List

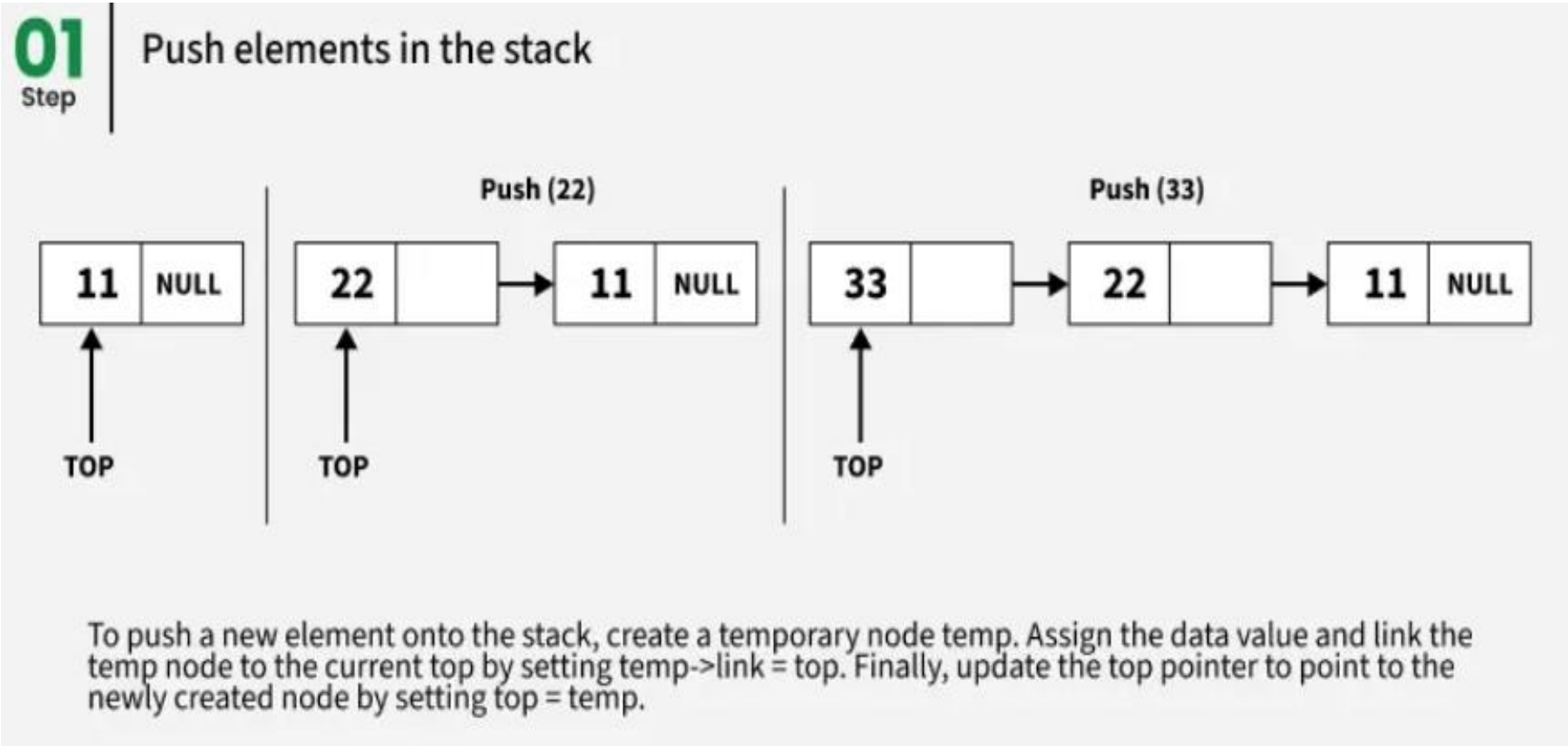
- In linked list implementation of stack, the nodes are maintained non-contiguously in the memory.
- Each node contains a pointer to its immediate successor node in the stack.
- Stack is said to be overflown if the space left in the memory heap is not enough to create a node.



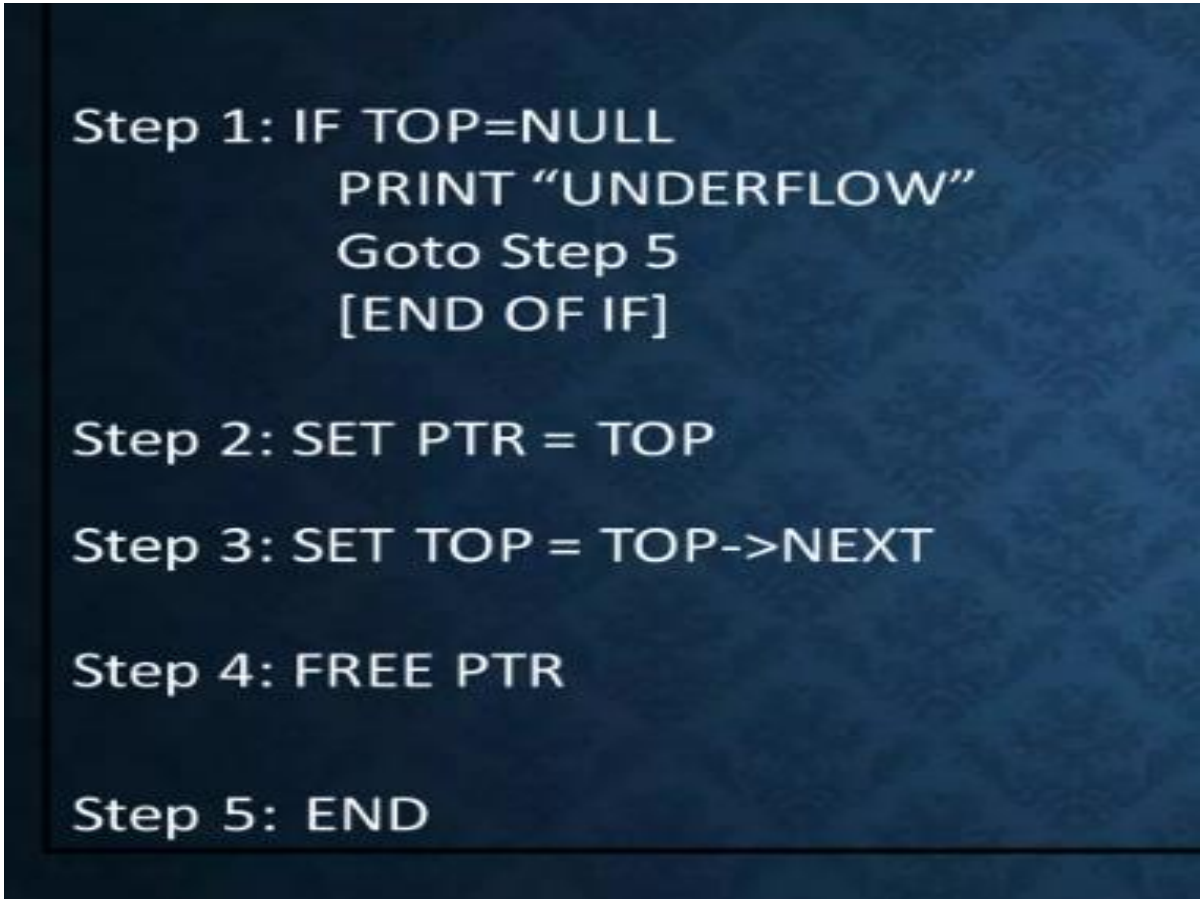
Adding a node to the stack (Push operation)

```
Step 1: Allocate memory for the new
        node and name it as NEW_NODE

Step 2: SET NEW_NODE -> DATA = VAL

Step 3: IF TOP = NULL
            SET NEW_NODE -> NEXT = NULL
            SET TOP = NEW_NODE
        ELSE
            SET NEW_NODE -> NEXT = TOP
            SET TOP = NEW_NODE
            [END OF IF]

Step 4: END
```

Adding a node to the stack (Push operation)

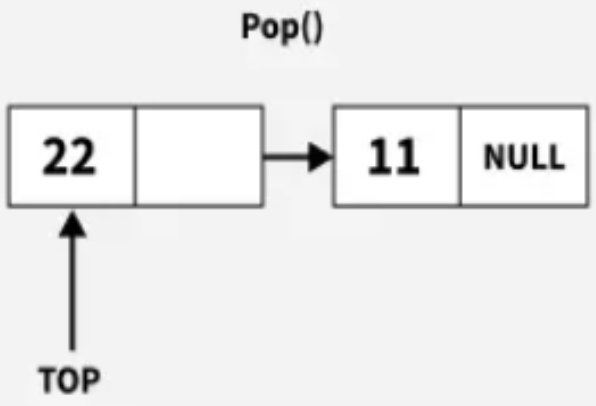


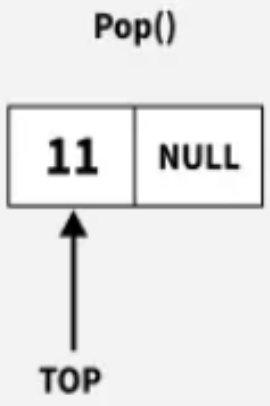
Deleting a node from the stack (POP operation)



Deleting a node from the stack (POP operation)

Pop elements from the stack






```
temp = top;
top = top → link;
temp → link = NULL;
free (temp);
```

Review

1.Operations: Push (adds to top), Pop (removes from top), Peek (returns top without removal), Is Empty, Size.

2.Implementation: Can be implemented using arrays, linked lists, or dynamic arrays (vectors in C++).

3.Complexity: O(1) time complexity for push, pop, peek, is Empty, and size operations.


THANK
YOU