ABES
Engineering College
College Code-032

Data Structure Training

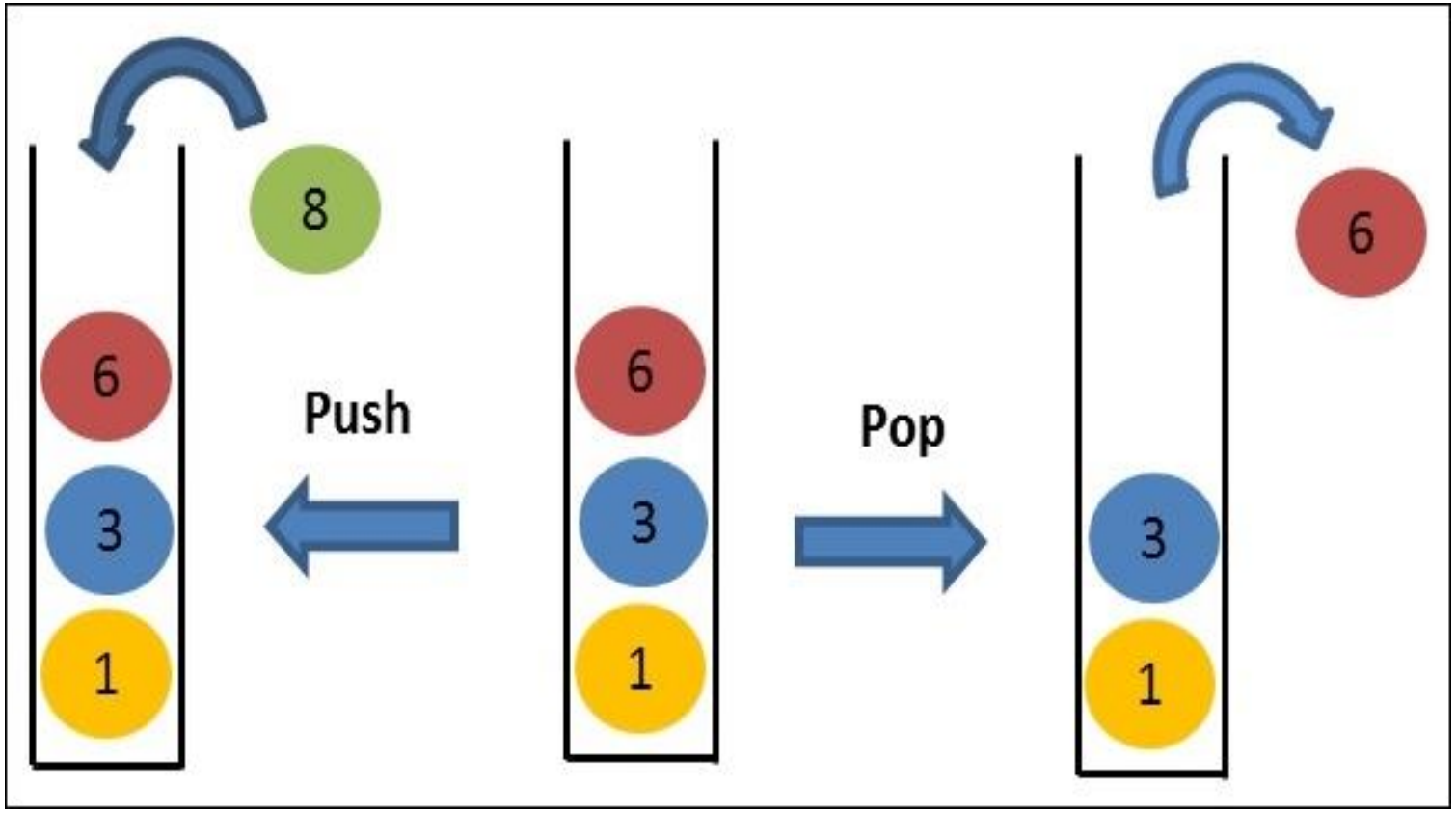
STACKS

Department of CSE,CEIT, CS & IT
ABES Engineering College, Ghaziabad

Data Structure Training

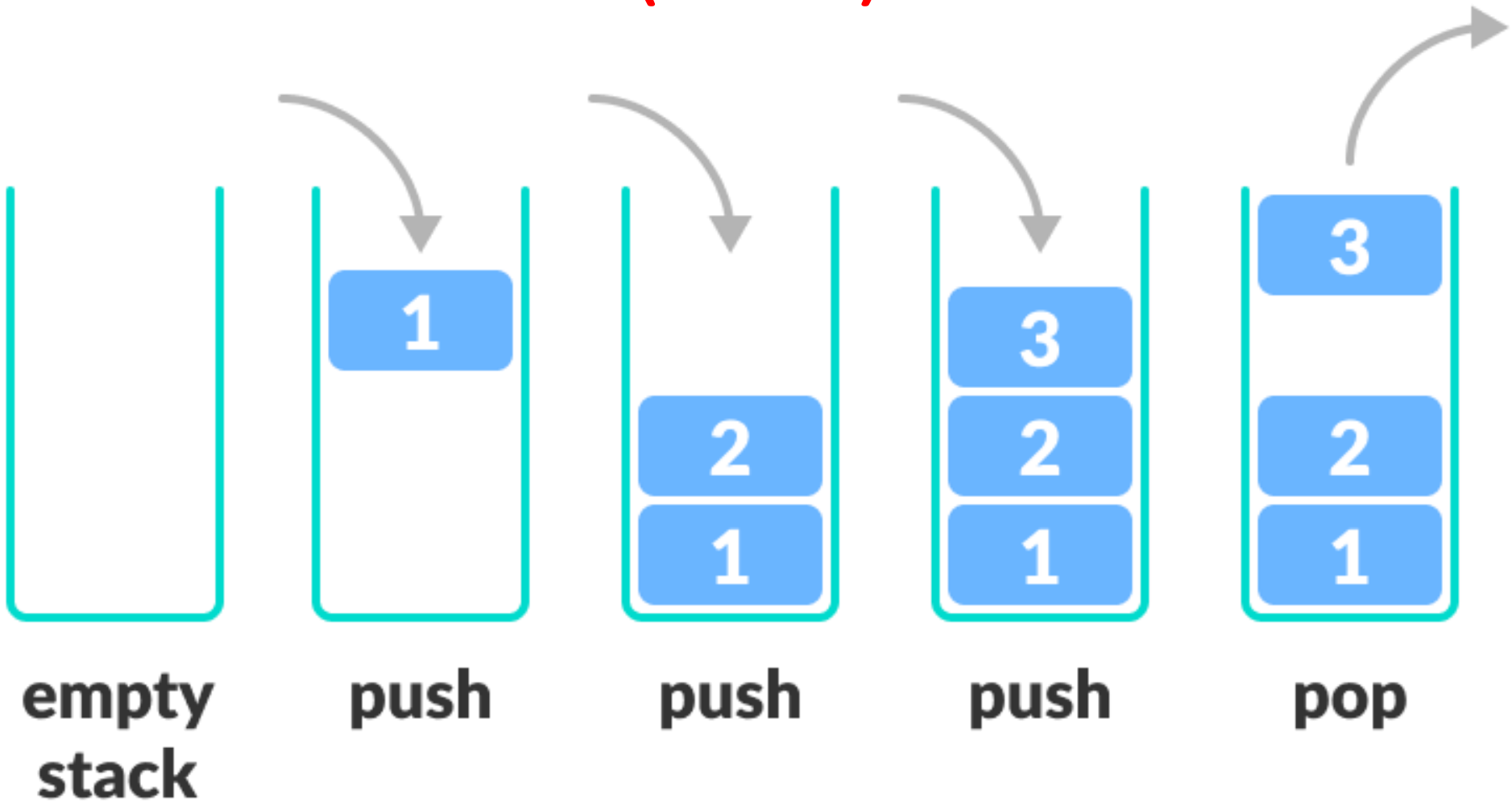
(Stack)

Basic operations on STACK



Data Structure Training

(Stack)



Data Structure Training

(Stack)

Implement Stack Using Array(Push)

```
#define MAX 5
int top;
int push(char Stack[], char x)
{
	if(top==MAX-1)
	{
		printf("stack overflow");
		exit(1);
	}
	else
	{
		Stack[++top]	= x;
		Stack[top++]	= x;
	}
}
```

Out of the 2 statements in else, only 1 is correct. Which one should be eliminated

- Red Coloured Statement
- Green Coloured Statement?

Data Structure Training

(Stack)

Implement Stack Using Array(Push)

```
#define MAX 5
int top;
int push(char Stack[], char x)
{
	if(top==MAX-1)
	{
		printf("stack overflow");
		exit(1);
	}
	else
	{
		Stack[++top]	= x;
		Stack[top++]	= x;
	}
}
```

Out of the 2 statements in else, only 1 is correct. Which one should be eliminated

- Red Coloured Statement
- Green Coloured Statement?

Data Structure Training
(Stack)

Implement Stack Using Array(pop)

```
#define MAX 5
int top;
char pop(char Stack[]){
	char x;
	if(top==-1)
	{
		printf("stack underflow");
		exit(1);
	}
	else
	{
		x = Stack[--top];
		x = Stack[top--];
	}
	return x;
}
```

Out of the 2 statements in else, only 1 is correct. Which one should be eliminated

- Red Coloured Statement
- Green Coloured Statement?

Data Structure Training

(Stack)

Implement Stack Using Array(pop)

```
#define MAX 5
int top;
char pop(char Stack[]){
	char x;
	if(top==-1)
	{
		printf("stack underflow");
		exit(1);
	}
	else
	{
		x = Stack[--top];
		x = Stack[top--];
	}
	return x;
}
```

Out of the 2 statements in else, only 1 is correct. Which one should be eliminated

- Red Coloured Statement

- Green Coloured Statement?

Data Structure Training

(Stack)

Consider the following operation performed on a stack of size 5.

Push(1);
Pop();
Push(2);
Push(3);
Pop();
Push(4);
Pop();
Pop();
Push(5);

After the completion of all operation, the no of element present on stack are

a) 1
b) 2
c) 3
d) 4

Data Structure Training

(Stack)

Consider the following operation performed on a stack of size 5.

Push(1);
Pop();
Push(2);
Push(3);
Pop();
Push(4);
Pop();
Pop();
Push(5);

After the completion of all operation, the no of element present on stack are

a) 1
b) 2
c) 3
d) 4

Data Structure Training
(Stack)

If the elements “A”, “B”, “C” and “D” are placed in a stack and are deleted one at a time, in what order will they be removed?

a) ABCD
b) DCBA
c) DCAB
d) ABDC

Data Structure Training
(Stack)

If the elements “A”, “B”, “C” and “D” are placed in a stack and are deleted one at a time, in what order will they be removed?

a) ABCD
b) DCBA
c) DCAB
d) ABDC

Data Structure Training

(Stack)

Sort a stack using a temporary stack

Input : [34, 3, 31, 98, 92, 23]

Output : [3, 23, 31, 34, 92, 98]

Create a temporary stack say **tmpStack**.

WHILE input stack is NOT empty DO

Pop an element from input stack call it **temp**

while temporary stack is NOT empty and top of temporary stack is greater than temp,

pop from temporary stack and push it to the input stack

push **temp** in temporary stack

The sorted numbers are in tmpStack

Data Structure Training

(Stack)

```
input: [34, 3, 31, 98, 92, 23]
 Element taken out: 23
 input: [34, 3, 31, 98, 92]
        tmpStack: [23]

Element taken out: 92
input: [34, 3, 31, 98]
        tmpStack: [23, 92]

Element taken out: 98
input: [34, 3, 31]
        tmpStack: [23, 92, 98]

Element taken out: 31
input: [34, 3, 98, 92]
        tmpStack: [23, 31]

Element taken out: 92
input: [34, 3, 98]
        tmpStack: [23, 31, 92]

Element taken out: 98
input: [34, 3]
        tmpStack: [23, 31, 92, 98]

Element taken out: 3
input: [34, 98, 92, 31, 23]
        tmpStack: [3]
```

Data Structure Training
(Stack)

```
Element taken out: 23
input: [34, 98, 92, 31]
       tmpStack: [3, 23]

Element taken out: 31
input: [34, 98, 92]
       tmpStack: [3, 23, 31]

Element taken out: 92
input: [34, 98]
       tmpStack: [3, 23, 31, 92]

 Element taken out: 98
input: [34]
       tmpStack: [3, 23, 31, 92, 98]

Element taken out: 34
input: [98, 92]
       tmpStack: [3, 23, 31, 34]

Element taken out: 92
input: [98]
       tmpStack: [3, 23, 31, 34, 92]

Element taken out: 98
input: []
       tmpStack: [3, 23, 31, 34, 92, 98]

final sorted list: [3, 23, 31, 34, 92, 98]
```

Data Structure Training
(Stack)

Sort the following:

Input : 8 5 7 1 9 12 10

Output : 1 5 7 8 9 10 12

Input : 7 4 10 20 2 5 9 1

Output : 1 2 4 5 7 9 10 20

Data Structure Training

(Stack)

Polish Notation and Reverse Polish Notation

Normal Notation:

INFIX → X+Y

Polish Notation :

PREFIX → +XY

Reverse Polish Notation :

POSTFIX → XY+

Data Structure Training
(Stack)

1. The postfix form of the expression (A+ B)*(C*D- E)*F / G is?

a) AB+ CD*E - FG /**
b) AB + CD* E - F **G /
c) AB + CD* E - *F *G /
d) AB + CDE * - * F *G /

Data Structure Training
(Stack)

1. The postfix form of the expression (A+ B)*(C*D- E)*F / G is?

a) AB+ CD*E - FG /**
b) AB + CD* E - F **G /
c) AB + CD* E - *F *G /
d) AB + CDE * - * F *G /

Data Structure Training
(Stack)

The prefix form of A-B/ (C * D ↑ E) is?

a) -/* ↑ ACBDE
b) -ABCD* ↑ DE
c) -A/B*C ↑ DE
d) -A/BC* ↑ DE

Data Structure Training
(Stack)

The prefix form of A-B/ (C * D ↑ E) is?

a) -/* ↑ ACBDE
b) -ABCD* ↑ DE
c) -A/B*C ↑ DE
d) -A/BC* ↑ DE

Data Structure Training
(Stack)

The prefix form of an infix expression p + q - r * t is?

a) + pq - *rt
b) - + pqr * t
c) - + pq * rt
d) - + * pqrt

Data Structure Training
(Stack)

The prefix form of an infix expression p + q - r * t is?

a) + pq - *rt
b) - + pqr * t
c) - + pq * rt
d) - + * pqrt

Data Structure Training
(Stack)

The result of evaluating the postfix expression
5, 4, 6, +, *, 4, 9, 3, /, +, * is?

a) 600
b) 350
c) 650
d) 588

Data Structure Training
(Stack)

The result of evaluating the postfix expression 5, 4, 6, +, *, 4, 9, 3, /, +, * is?

a) 600

b) 350

c) 650

d) 588

Data Structure Training
(Stack)

Number conversion using STACK

The decimal number (233)10 and its corresponding binary equivalent (11101001)2 are interpreted respectively as

$$2 \times 10^2 + 3 \times 10^1 + 3 \times 10^0$$

and

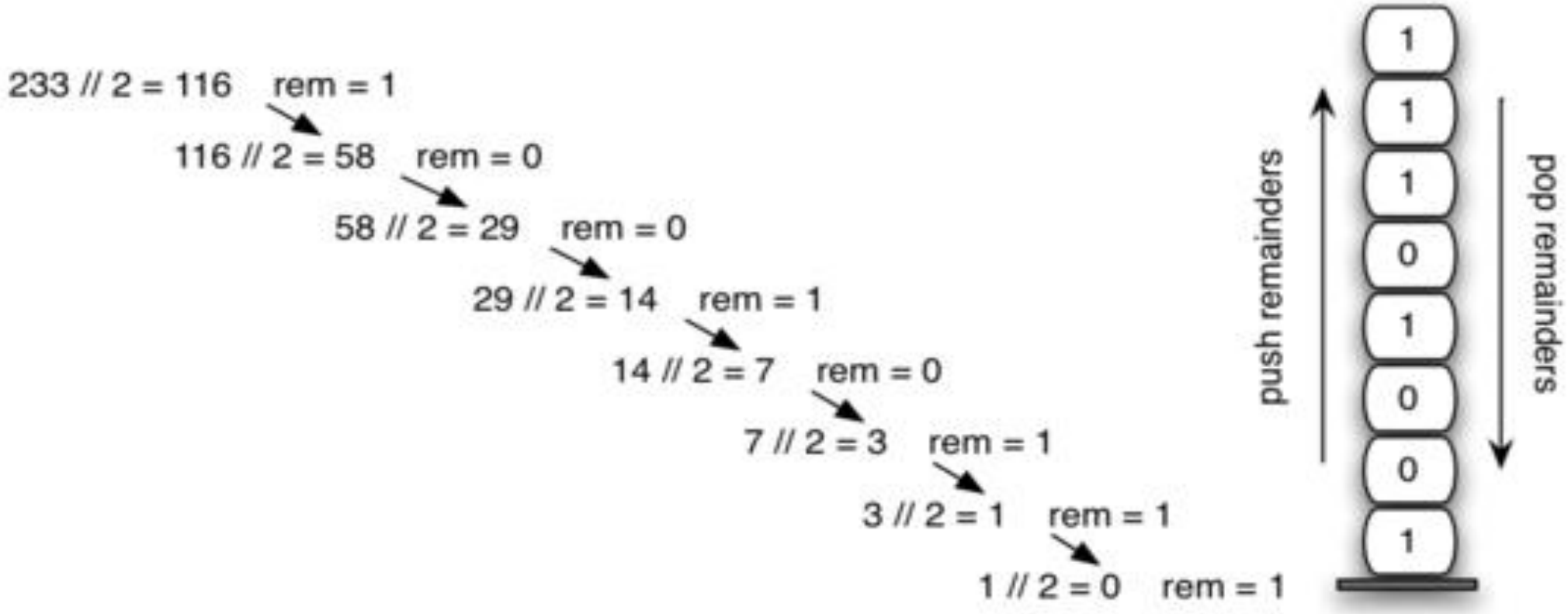
$$1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

But how can we easily convert integer values into binary numbers?
The answer is an algorithm called “Divide by 2” that uses a stack to keep track of the digits for the binary result.

Data Structure Training

(Stack)

Conversion of Decimal to Binary using STACK



Data Structure Training
(Stack)

Convert 98_{10} to Binary

a) 1100110_2

b) 1110010_2

c) 1110011_2

d) 1101011_2

e) 1100010_2

Data Structure Training
(Stack)

Convert 98_{10} to Binary

a) 1100110_2

b) 1110010_2

c) 1110011_2

d) 1101011_2

e) 1100010_2

	98	
2	49	-0
2	24	-1
2	12	-0
2	6	-0
2	3	-0
2	1	-1

Data Structure Training
(Stack)

Convert 456_{10} to Binary

a) 111001001

b) 111001100

c) 111001010

d) 111001000

e) 110001000

Data Structure Training

(Stack)

Convert 456_{10} to Binary

a) 111001001

b) 111001100

c) 111001010

d) 111001000

e) 110001000

	456	
2	228	-0
2	114	-0
2	57	-0
2	28	-1
2	14	-0
2	7	-0
2	3	-1
2	1	-1

Data Structure Training
(Stack)

Consider the following pseudo code that uses a stack. What is output for input "letsfindc"?

```
declare a stack of characters
while ( there are more characters in the word to read )
        read a character push the character on the stack
while ( the stack is not empty )
        pop a character off the stack write the character to the screen
```

A. letsfindcletsfindc
B. cdnifstel
C. letsfindc
D. cdnifstelcdnifstel

Data Structure Training

(Stack)

Consider the following pseudo code that uses a stack. What is output for input "letsfindc"?

```
declare a stack of characters
while ( there are more characters in the word to read )
        read a character push the character on the stack
while ( the stack is not empty )
        pop a character off the stack write the character to the screen
```

A. letsfindcletsfindc

B. **cdnifstel**

C. letsfindc

D. cdnifstelcdnifstel

Data Structure Training
(Stack)

String operations using STACK

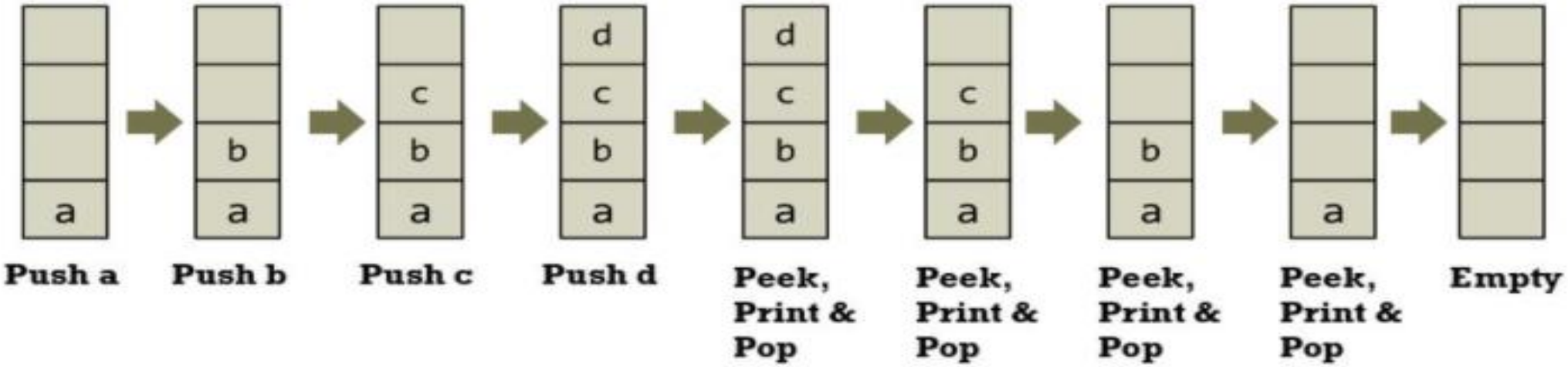
- Reverse a string
- Reverse alphabets of the words of a string

Data Structure Training
(Stack)

Reverse a string

1. Create an empty stack.
2. One by one push all characters of string to stack.
3. One by one pop all characters from stack and put them back to string.

Input: abcd
Output: dcba



Data Structure Training
(Stack)

Reverse the words of a string

1. Create an empty stack.
2. Traverse the entire string, while traversing add the characters of the string into a temporary
   variable until you get a space(‘ ‘) and push that temporary variable into the stack.
3. Repeat the above step until the end of the string.
4. Pop the words from the stack until the stack is not empty which will be in reverse order.

- ***Input:*** *str = “data structures and algorithms”*
  Output: *algorithms and structures data*

Data Structure Training

(Stack)

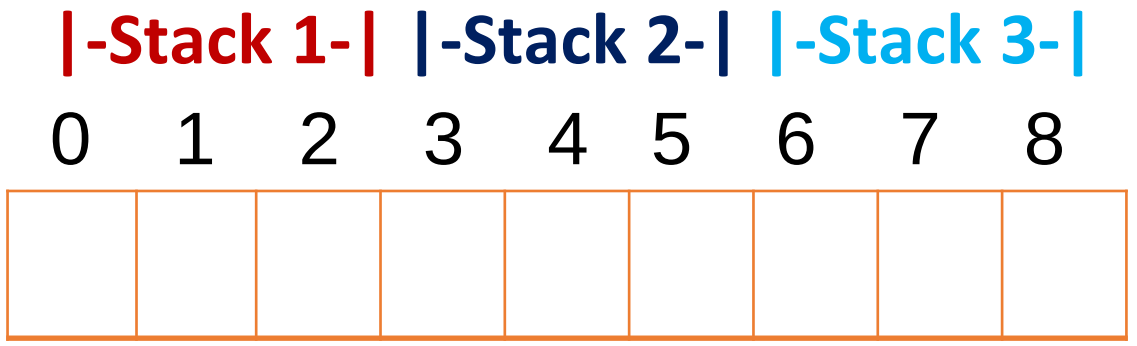
Solve the problem here:

https://leetcode.com/problems/reverse-string/

Data Structure Training
(Stack)

Implement Multiple Stack In Single Array

- A single array will store multiple stacks.
- For each stack in that array, there will be a top variable.
- let array size N=9
- number of stack M=3
- stack size= N/M
  =9/3=3
- if N=8
- Then stack size=2.6
- i.e. 3,3,2



Data Structure Training
(Stack)

Calculate Initial top of any Stack

Initially for the whole array it is -1

For 0^{th} stack, initial top of stack= 0 * 3 -1 =-1

For 1^{st} stack, initial top of stack= 1 * 3 -1 = 2

For 2^{nd} stack, initial top of stack= 2 * 3 -1 = 5

For i^{th} stack, initial top of stack= i * N/M -1

Data Structure Training
(Stack)

Multiple Stack(Push)

```
#define N 100
#define M 10
void push(char Stack[],int Ti, int i, char x)//insertion to stack i
{
	char x;
	if(--------------------)
	{
		printf("stack overflows");
		exit(1);
	}
	else
	{
		------------;
		------------;
	}
}
```

Data Structure Training
(Stack)

Multiple Stack(Push)

```
#define N 100
#define M 10
void push(char Stack[],int Ti, int i, char x)//insertion to stack i
{
	char x;
	if(Ti== (i+1)*N/M -1)
	{
		printf("stack overflow");
		exit(1);
	}
	else
	{
		------------;
		------------;
	}
}
```

Data Structure Training
(Stack)

Multiple Stack(Push)

```
#define N 100
#define M 10
void push(char Stack[],int Ti, int i, char x)//insertion to stack i
{
	char x;
	if(Ti== (i+1)*N/M -1)
	{
		printf("stack overflow");
		exit(1);
	}
	else
	{
		Ti++;
		stack[Ti]=x;
	}
}
```

Data Structure Training
(Stack)

Multiple Stack(pop)

```
#define N 100
#define M 10
char pop(char Stack[],int Ti, int i)//deletion from stack i
{
	char x;
	if(---------------)
	{
		printf("stack underflow");
		exit(1);
	}
	else
	{
		---------------;//store and then decrement
		---------------;
	}
	return x;
}
```

Data Structure Training
(Stack)

Multiple Stack(pop)

```
#define N 100
#define M 10
char pop(char Stack[],int Ti, int i)//deletion from stack i
{
	char x;
	if(Ti== i*N/M -1)
	{
		printf("stack underflow");
		exit(1);
	}
	else
	{
		--------------;//store and then decrement
		--------------;
	}
	return x;
}
```

Data Structure Training
(Stack)

Multiple Stack(pop)

```
#define N 100
#define M 10
char pop(char Stack[],int Ti, int i)//deletion from stack i
{
	char x;
	if(Ti== i*N/M -1)
	{
		printf("stack underflow");
		exit(1);
	}
	else
	{
		x=stack[Ti];//store and then decrement
		Ti--;
	}
	return x;
}
```

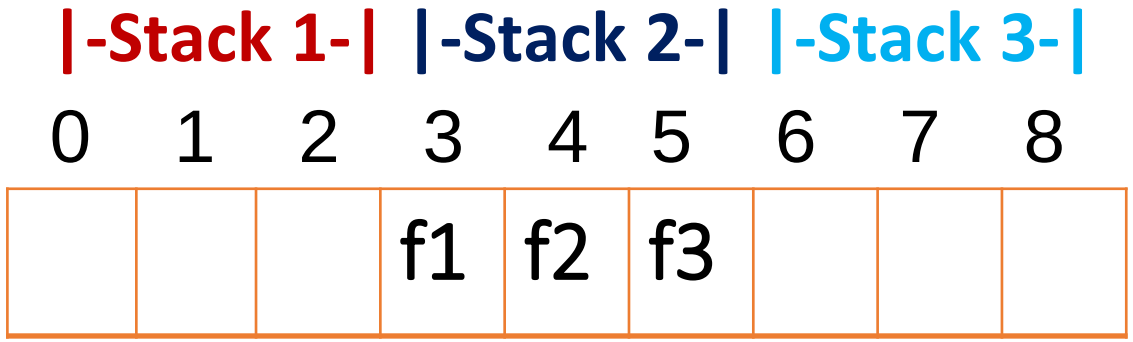
Data Structure Training
(Stack)

Advantage of Multiple Stack single array(MSSA)

If There is a single stack only in a single array,

- you can not run two recursive program at a time.
- By using MSSA concept ,you can run two recursive program at a time

Drawback of Multiple Stack single array(MSSA)

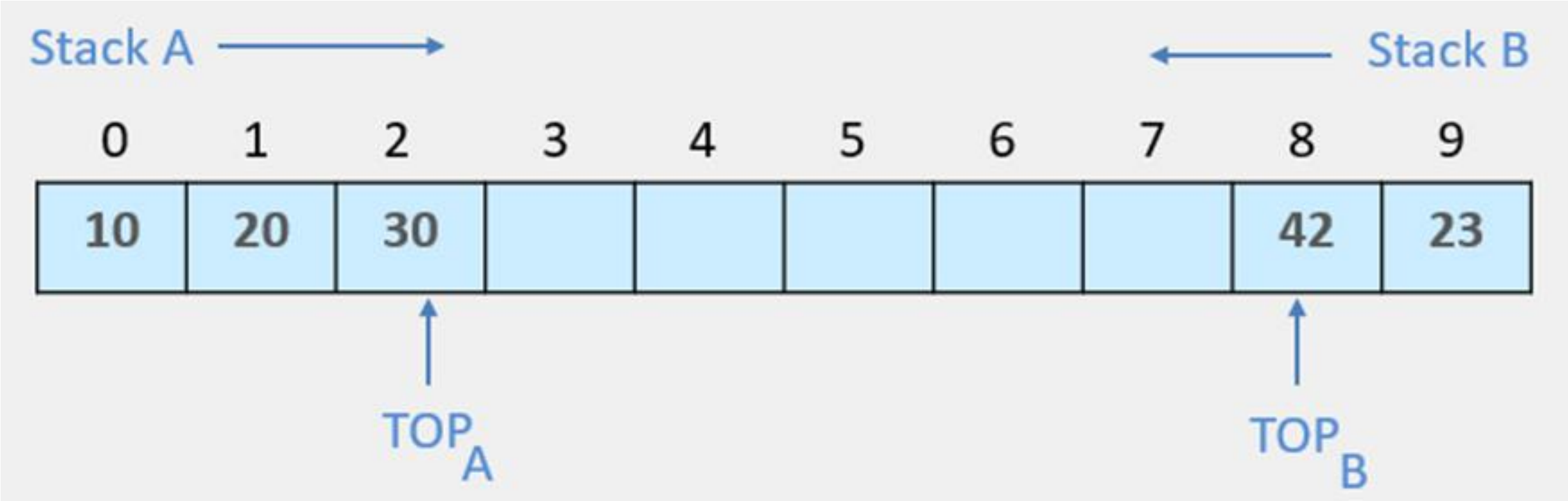


In above stack we can easily say that it is not efficient as a lot of space is empty (stack1 and stack3) but still get a message stack is full in case of stack2

Data Structure Training

(Stack)

Multi-Stack



Data Structure Training
(Stack)

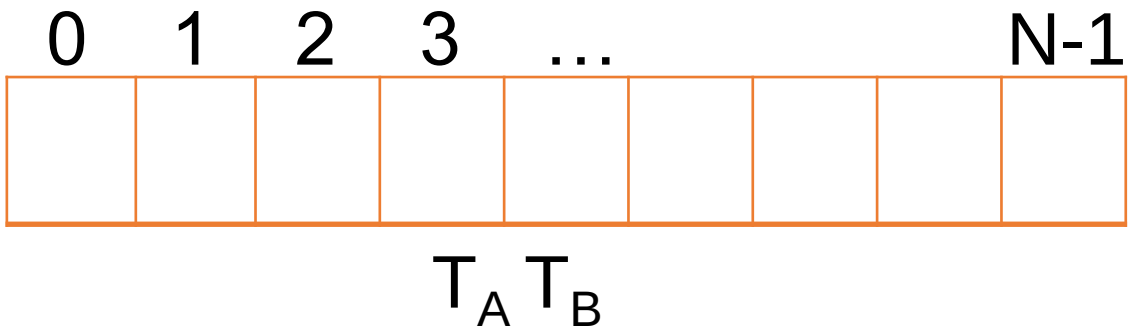
Multi-Stack

- A single stack is sometime not sufficient to store large amount of data.
- To overcome this problem we can use multiple stack.
- For this, we have used a single array having more than one stack.
- The array is divided for multiple stacks.
- Let there is an array **STACK[n]** divided into two stacks **STACK A**, **STACK B**, where **n = 10**.
- **STACK A** expands from the left to right, i.e. from 0th element.
- **STACK B** expands from the right to left, i.e, from 10th element.
- The combined size of both **STACK A** and **STACK B** never exceed 10.

Data Structure Training
(Stack)

Implement Multiple Stack in single array efficiently-

- Here memory is shared between two different stacks , which leads to the synchronization problem and that memory must be considered as critical section which must be handled by semaphore.
- An array is full when $T_A = T_B - 1$
- Stack empty when $T_A = -1$ & $T_B = N$
- In this strategy one stack will start from left and other will start from right.
- For insertion left will increment and right will decrement



Data Structure Training

(Stack)

Problems asked in placements

- Write a program to find out minimum number of an array using stack.
- https://practice.geeksforgeeks.org/problems/rotten-oranges2536/1
- Given an expression string **x**. Examine whether the pairs and the orders of “{“,”}”,”(“,”)”,”[“,”]” are correct in exp. For eg: {([])} would return true and ([ ] would return false.
- Given an input stream of **N** characters consisting only of lower case alphabets. The task is to find the first non repeating character, each time a character is inserted to the stream. If no non repeating element is found print -1.

Data Structure Training
(Stack)

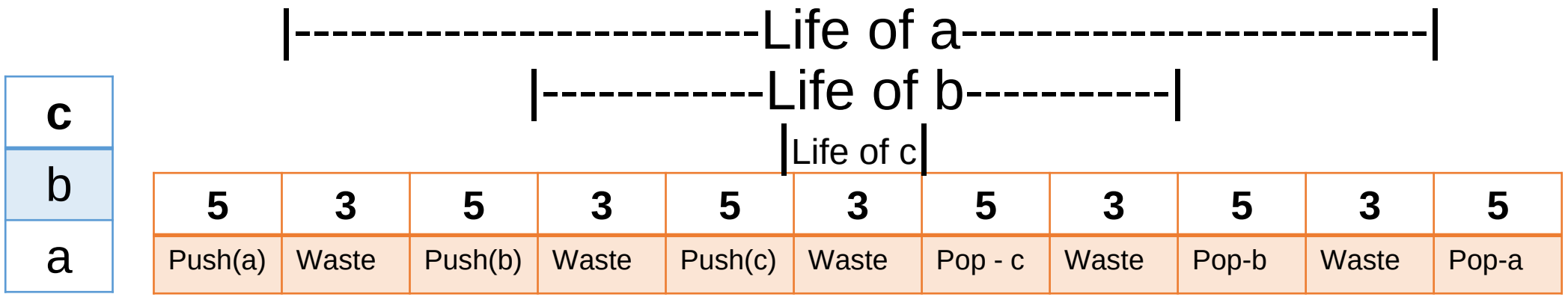
Stack Life Time of an Element

- life time of an element in stack is time after push operation of that element before pop operation of that element.
- Example1- Let us suppose that three elements (a,b,c) continuously pushed followed by pop.
- Let us assume that push() and pop() operation takes 5 min time.\
- Assume time wasted between any two operation is 3.

Data Structure Training
(Stack)

Stack Life Time of an Element

- life time of an element in stack is time after push operation of that element before pop operation of that element.
- Example1- Let us suppose that three elements continuously pushed followed by pop.
- Let us assume that push() and pop() operation takes 5 min time.\
- Assume time wasted between any two operation is 3.
- Life of c= 3, Life of b= 19, Life of a= 35
- Average Life = (3+19+35)/3




Thank
you