B.Tech.

GATEWAY CLASSES

Data Structure

Unit -2 : Lec- 1

CS & CS Allied

GATEWAY CLASSES

UNIT 2
1. Stack
2. Queue
3. Recursion & Iteration

Today's Target

- What is Stack Data Structure?
- Insertion and Deletion Operations on Stack i.e. PUSH and POP Operations on Stack Data Structure
- Algorithms: PUSH & POP Operations
- Representation of Stack: Static and Dynamic
- Implementation of PUSH and POP in C
- AKTU PYQs

GATEWAY CLASSES

What is Stack Data Structure?

- We know that linked list is a collection of nodes and there are different types of possible operations which can be performed on linked list data structure.
- Any item can be inserted to any position of linked list and in the same manner any item can be deleted from any position of the linked list.
- So in case of linked list which is a dynamic data structure, there is no restriction to perform any operation.
- Stack is another non primitive data structure in which it is possible to create the restriction for performing the operation on the data structure.
- Stack works on LIFO property.

GATEWAY CLASSES

..What is Stack Data Structure?

LIFO

- Stack is a linear data structure in which operation of insertion and deletion can be performed only at one end and that end is known as top of the stack.

TOS / TOP

- A common model of a stack is plates in a marriage party. Fresh plates are pushed onto the top and popped from the top.

remove

insert

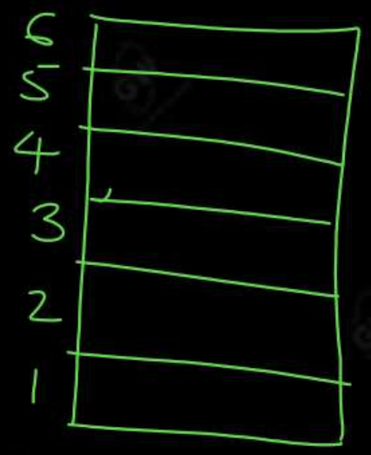
- The operation of insertion of an element is known as PUSH operation and the operation of deletion of an element is known as POP operation.

GATEWAY CLASSES

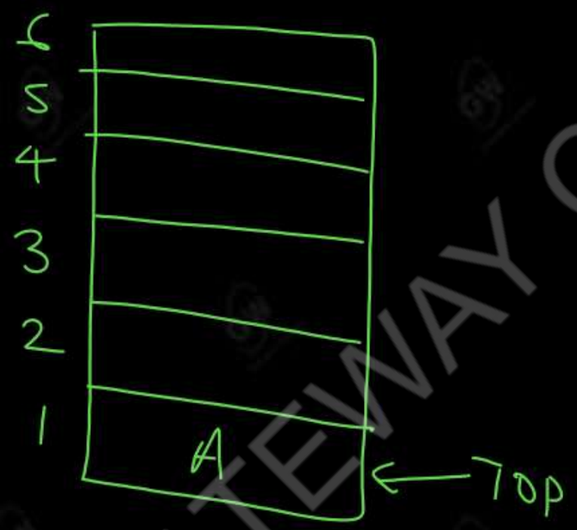
Example.1 :- Consider the following example where we have an empty STACK which can contain the following 6 elements:-

Elements are :- A, B, C, D, E, F

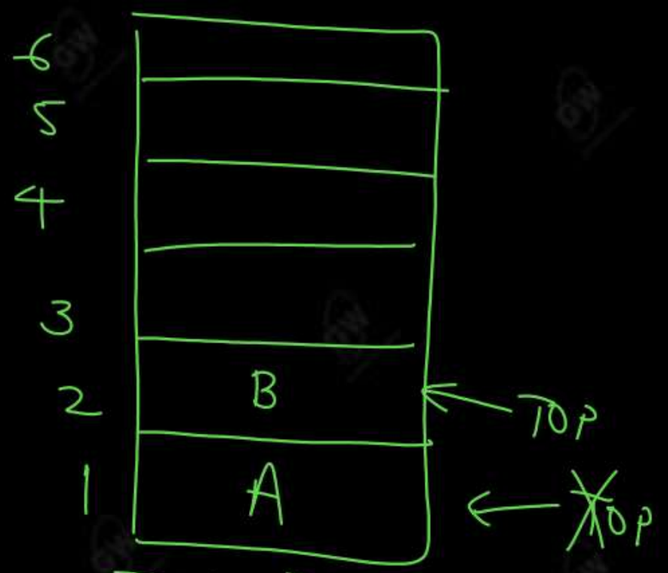
After pushing elements one by one the STACK will be as follows:-



STACK: Initial Empty
TOS = 0
TOP = 0
(a)

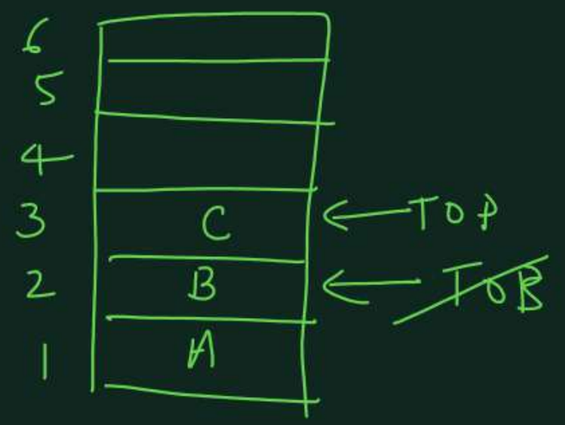


STACK: PUSH 'A'
OR STACK[TOP] = PUSH 'A'
TOP = TOP + 1 = 0 + 1 = 1
(b)



TOP = TOP + 1 = 1 + 1 = 2
STACK[TOP] = PUSH 'B'
(c)

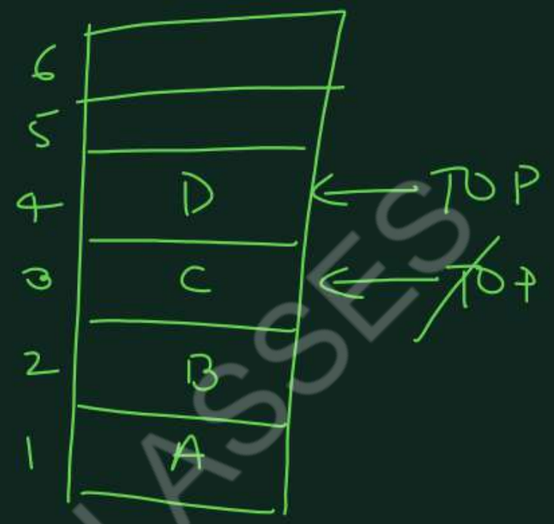
GATEWAY CLASSES



TOP++; TOP = TOP+1
= 2+1
= 3

STACK[TOP] = PUSH 'C'

(d)



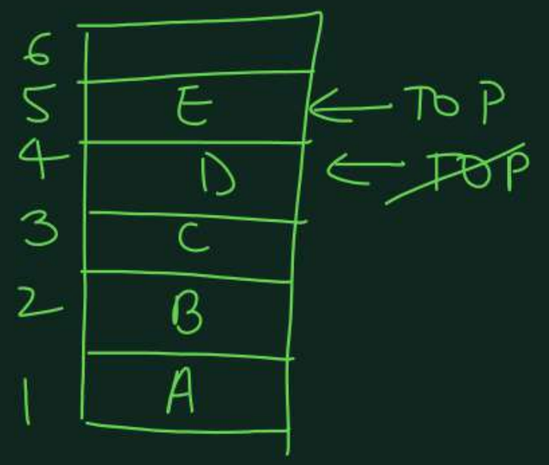
TOP++; TOP = TOP+1
= 3+1 = 4

STACK[TOP] = PUSH 'D'

STACK[4] = 'D'

(e)

GATEWAY CLASSES

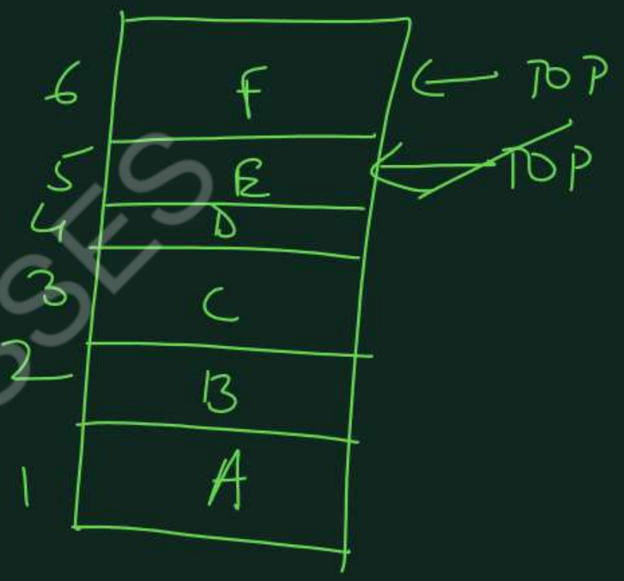


TOP++ ; TOP = TOP+1
= 4+1=5

STACK[TOP] = PUSH 'E'

STACK[5] - E

(f)



TOP++ ; TOP = TOP+1
= 5+1=6

STACK[TOP] = PUSH 'F'

i.e. STACK[6] = 'F'

(g)

GATEWAY CLASSES

Example.2 :- Consider the following STACK of characters where STACK can contain 10 items i.e. 10 memory cells are available.

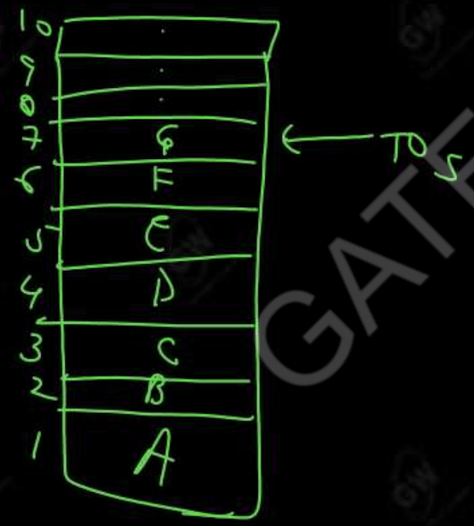
STACK:- A, B, C, D, E, F, G, -, -, -

Describe the STACK as the following operations takes place. Here PUSH means insertion of an element and POP means deletion of an element. Write the final STACK after performing all operations. List the popped items.

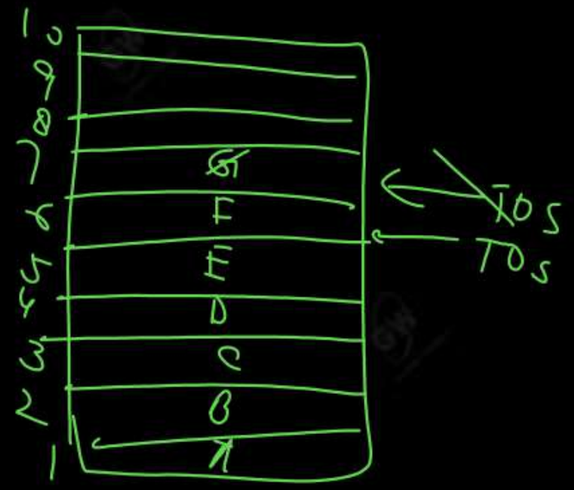
(a) POP(STACK,ITEM)
(b) PUSH(STACK,H)
(c) POP(STACK,ITEM)
(d) PUSH(STACK,K)
(e) POP(STACK,ITEM)
(f) PUSH(STACK,I)
(g) PUSH(STACK,J)
(h) POP(STACK,ITEM)

Solution:- 10 items

Initial Stack
TOS = 7



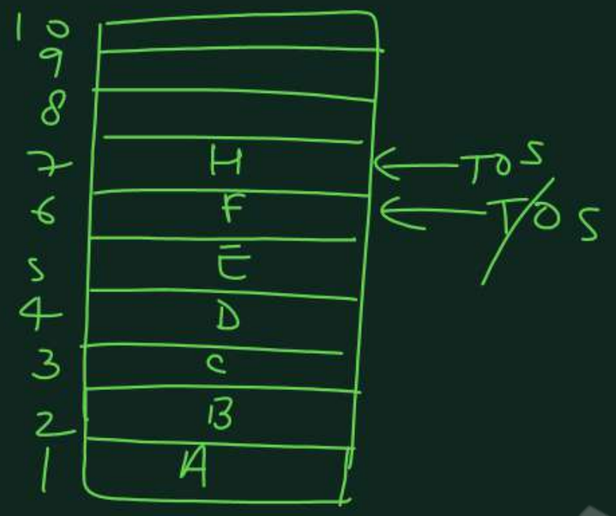
(a)



ITEM = STACK[TOS]
= STACK(7)
ITEM = 'G'
TOS = TOS − 1
= 7 − 1
= 6

GATEWAY CLASSES

(b) PUSH (STACK, H)

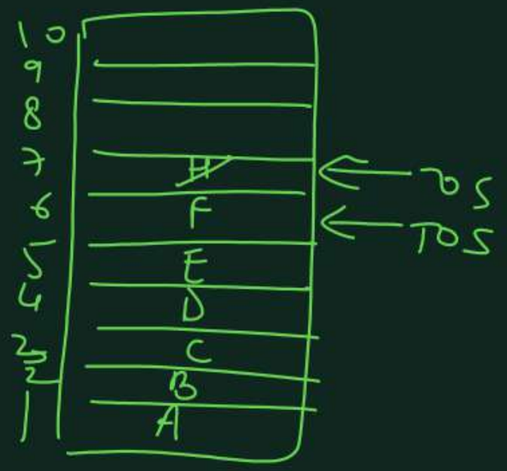


TOS = TOS + 1
= 6 + 1
= 7

STACK[TOS] = ITEM

STACK[7] = H

(c) POP (STACK, ITEM)



ITEM = STACK[TOS]

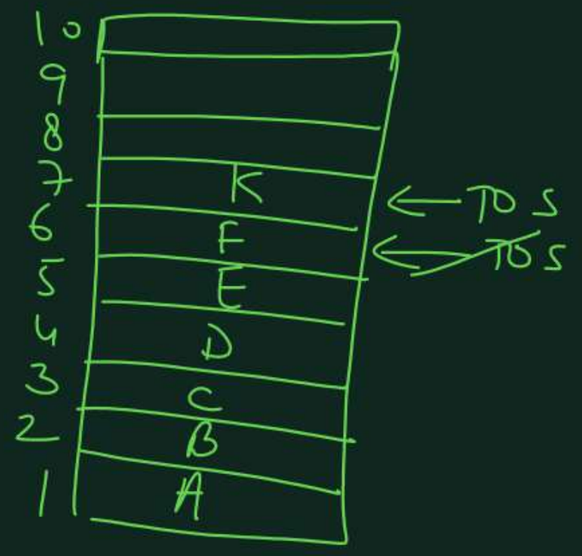
ITEM = STACK[7]
= H

TOS = TOS - 1

TOS = 7 - 1 = 6

GATEWAY CLASSES

(d) PUSH (STACK, K)

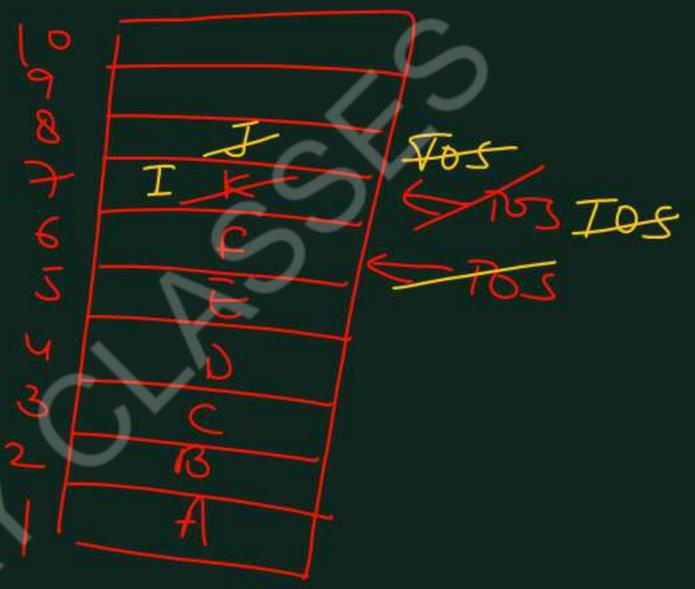


STACK

TOS++

STACK[TOS] = ITEM
= K

(c) POP (STACK, ITEM)



ITEM = STACK[TOS]
= K
TOS = TOS − 1
= 7 − 1 = 6

item = 'J'

PUSH
POP } TOS

GATEWAY CLASSES

PUSH, POP — TOS

PUSH — you should have the item.

POP — TOP Element → Item, TOS = TOS − 1

Stack full

N = 10

TOS = 10

PUSH " OVERFLOW "

Stack empty

TOP = ≯ 0

POP " UNDERFLOW "

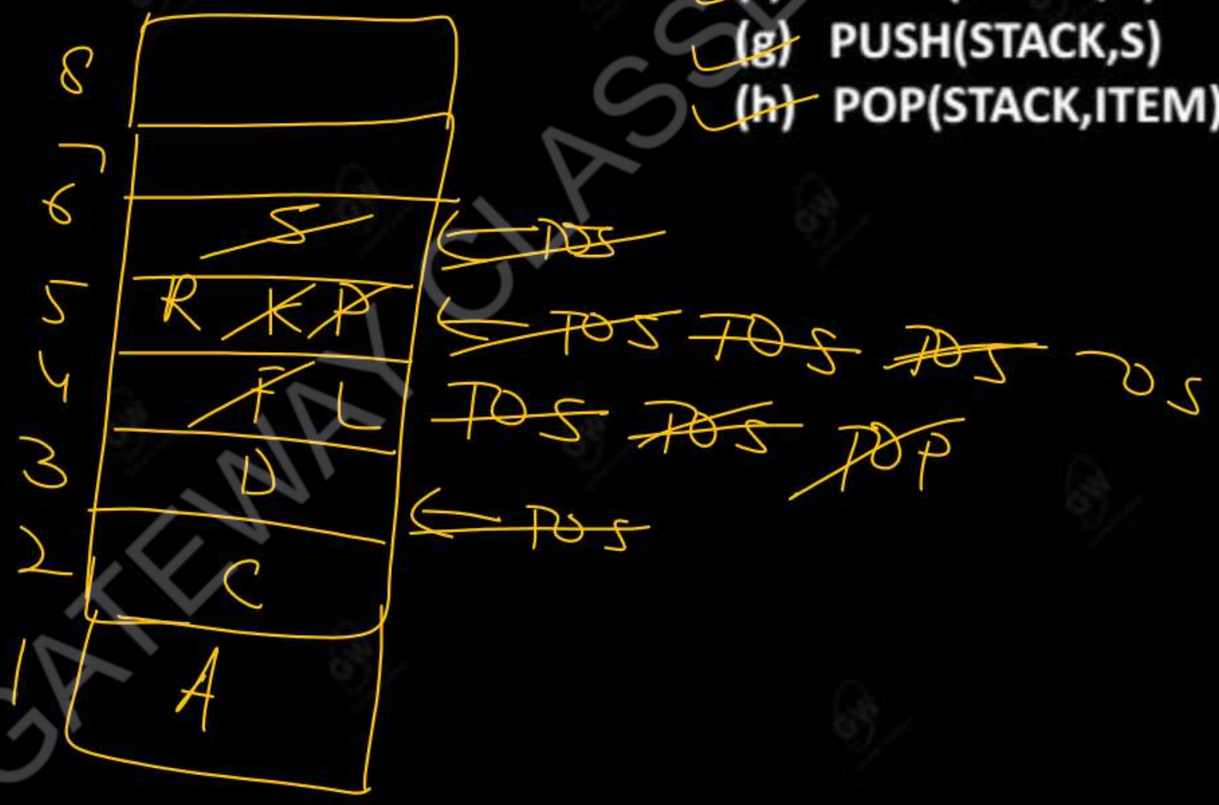
GATEWAY CLASSES

Example.3 :- Consider the following STACK of characters, where STACK is allocated, N=8 memory cells.

STACK; A, C, D, F, K, -, -, -

Describe the STACK as the following operations takes place-

(a) POP(STACK,ITEM)
(b) POP(STACK,ITEM)
(c) PUSH(STACK,L)
(d) PUSH(STACK,P)
(e) POP(STACK,ITEM)
(f) PUSH(STACK,R)
(g) PUSH(STACK,S)
(h) POP(STACK,ITEM)

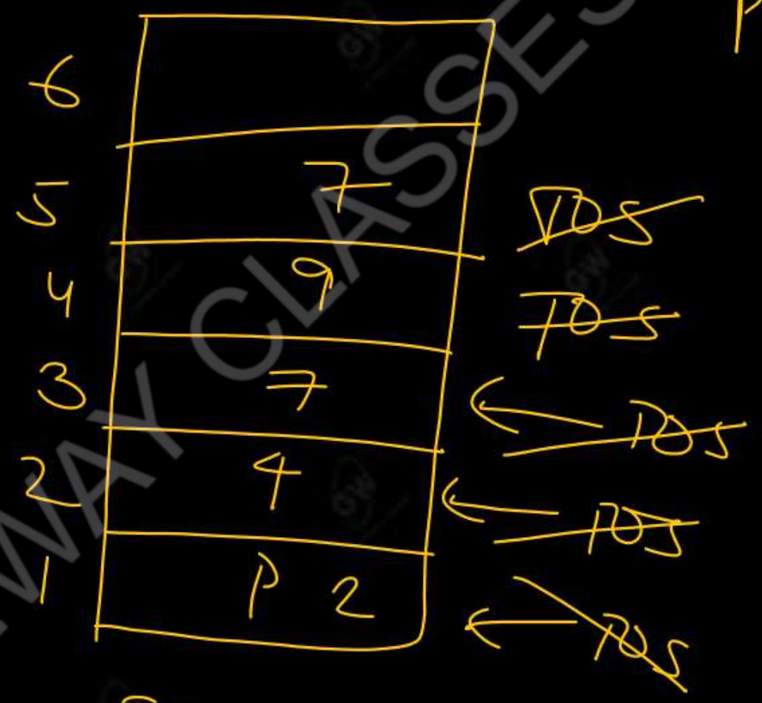


GATEWAY CLASSES

EXAMPLE. 6 :- Suppose STACK is allocated N=6 memory cells and initially STACK is empty, or, in other words, TOP=0. Find the output of the following module:

1. Set P = 2 and Q = 5.
2. Call PUSH(STACK,P)
Call PUSH(STACK,4)
Call PUSH(STACK,Q+2)
Call PUSH(STACK,9)
Call PUSH(STACK,P+Q).
3. Repeat while TOP=0;
Call POP(STACK,DATA)
Write : DATA.
[End of Loop]
4. Return.

N=6



STACK

TOP=0

P=2, Q=5

5+2=7

7, 9, 7, 4, 2

Ans

GATEWAY CLASSES

Representation of Stack

A stack can be implemented by 2 ways:-

1. Static implementation of Stack
2. Dynamic implementation of Stack (linked list)

1. Static implementation of Stack:- Stack can be implemented in memory by using arrays. In this technique of stack implementation, the size of the stack (array) is defined during design time. It is not the efficient way, in terms of memory usage. At run time, there is no way to increase or decrease the size of the array.

2. Dynamic Implementation of stack:- Dynamic implementation of stack is possible by using the concept of linked list data structure. It is a dynamic data structure where the size of data structure can be increased or decreased very easily at run time.

GATEWAY CLASSES

Possible Operations on Stack:

Two important operations which can be performed on Stack data structure are as follows:-

1. PUSH
2. POP

Array

no be inserted

top of the stack

ALGORITHM:- PUSH (STACK, ITEM, TOS, MAXSIZE) → Array, fixed, maxsize

This algorithm is used to insert an element ITEM to the top of the stack (TOS) of STACK which is of size MAXSIZE.

Step 1: - [Check for the Overflow]

(i) if (TOS == MAXSIZE – 1) F

(ii) {

STACK

(iii) Write " Overflow" or "Insertion is not possible" or "PUSH operation is not possible".

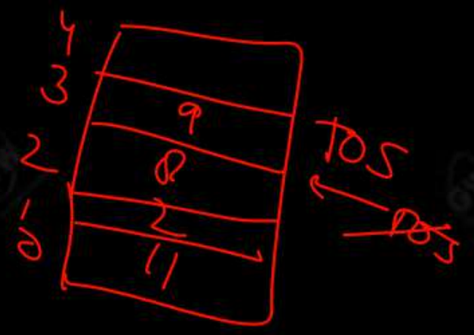
(iv) exit

(v) }

Step 2:- TOS = TOS + 1.

Step 3:- STACK [TOS] = ITEM.

Step 4:- Stop



GATEWAY CLASSES

ALGORITHM:- POP (STACK, ITEM, TOS)

This algorithm is used to delete an element ITEM from the top of the stack (TOS) of STACK.

Step 1: - [Check for Underflow]

(i) if (TOS == - 1)

(ii) {

(iii) Write " Underflow" or "Deletion is not possible" or "POP operation is not possible".

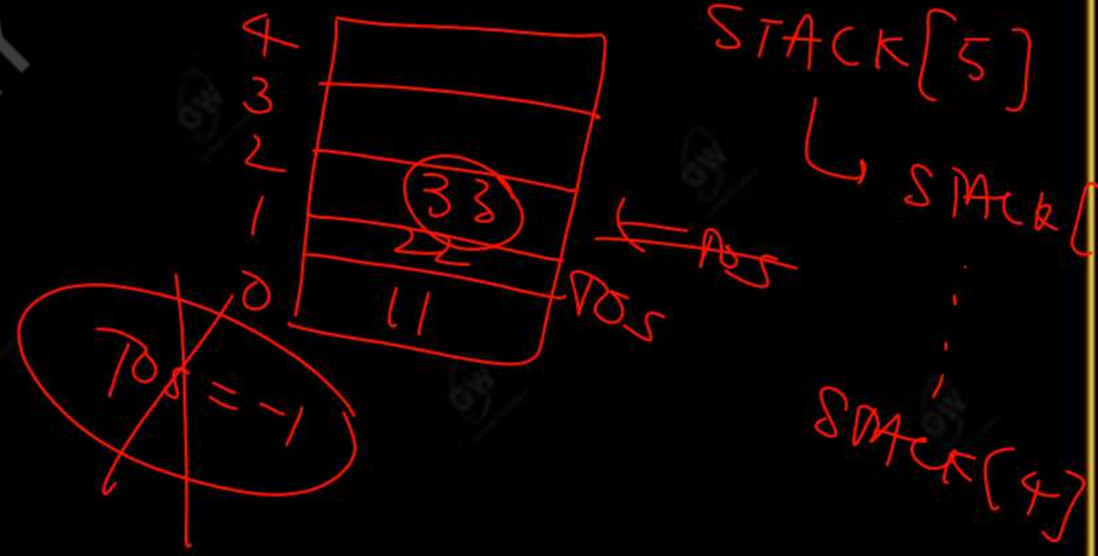
(iv) exit

(v) }

Step 2:- ITEM = STACK [TOS]

Step 3:- TOS = TOS - 1.

Step 4:- Stop



GATEWAY CLASSES

Push and Pop Operations Implementation in C or Array

Implementation of Stack in C

```
#include<stdio.h>
int top=-1;
int stack[20];
void push (int *, int, int);
int pop ( int * );
void display ( int *);
void main()
{
int data, ch,size;
clrscr();
printf("\n enter the size of stack:");
scanf("%d",&size);
if(size>20)
{
printf("\n Invalid size, it can not be grater than
20");
getch();
exit();
}
do
{
printf(" \n\n Available options are as follows:");
printf("\n\n \t Press 1 to Push an element to the
stack");
printf("\n\n \t Press 2 to Pop the element from the
stack");
printf("\n\n \t Press 3 to Quit from the
application");
printf("\n\n enter your choice:");
scanf("%d",&ch);
```

/* array of 20 elements */

FP

GATEWAY CLASSES

```
switch(ch)
{
case 1:
printf("\n\n enter the element to push to
stack:");
scanf("%d",&data);
push( stack, size, data);
printf("\n\n after performing the Push
operation the stack is as follows:");
display(stack);
break;
case 2:
data=pop(stack);
printf("\n\n the popped data is:%d",data);
printf("\n\n after performing Pop operation the
stack is as follows:");
display(stack);
break;
case 3:
exit();
}
}while(ch!=3);
getch()
}
```

/* funⁿ call */

GATEWAY CLASSES

```c
/* Push Operation */
void push (int *S, int size, int d)
{
if(top== size - 1)
printf("\n Insertion is not possible due to stack full i.e. Stack Overflow");
else
{
top=top+1;
S[top]=d;
}
}
/* Pop Operation */
int pop(int *S)
{
int popped_element;
if(top==-1)
{
printf("\n Deletion is not possible i.e. Stack Underflow");
return 0;
}
else
{
popped_element = S[top];
top--;
}
return popped_element;
}
/* Display Operation */
void display(int s[])
{
int i;
if(top==-1)
printf("\n\n Stack is empty");
else
{
for(i=top;i>=0;i--)
printf("\n %d",s[i]);
}
}
```

GATEWAY CLASSES

PYQs

Q.1 Write the Push an POP functions in C simulating Push and Pop operations of STACK implemented using an array of integers. (AKTU 2014-15, 5 Marks)

Q.2 Write algorithms for Push and Pop operations in stack. Transform the following expression into its equivalent postfix expression using stack:

A + (B * C - (D / E ^ F) * G) * H (AKTU2019-20, 10 Marks)

GATEWAY CLASSES

Download Gateway Classes Application
From Google Play store
Link in Description

Next Lecture:- Some Major Applications of Stack Data Structure, Evaluation of Arithmetic Expression, Notation Conversions

Thank You

GATEWAY CLASSES

B.Tech

GATEWAY CLASSES

Data Structure

Unit -2 : Lec-2

CS & CS Allied

GATEWAY CLASSES

Unit-2

1. Stack
2. Queue
3. Recursion & iteration

Today's Target

- Some Major Applications of Stack Data Structure
- Evaluation of Arithmetic expression
- Notations Conversion
- AKTU PYQs

Prefix, infix, postfix

-AB A+B AB+

GATEWAY CLASSES

Some Major Applications of Stack Data Structure

There are number of applications of stacks, some of them are as follows:-

1. Stack Framing:-

- Almost all high level languages uses the concept of stack framing. C program use the concept of stack framing for working memory of each function calling. When any function is called the stack framing is done and the data is pushed on to the top of the stack. When the function returns, the data is pooped off the stack.

When a function call to itself

- To implement the concept of recursion this technique is also very useful. The stack is a region of main memory within which programs temporarily stored data at the time of its execution. For example when a programs sends parameters to a function, the parameters are placed on to the stack. When the function completes its execution these parameters are popped off from the stack.
- When a function calls other function the current contents of the caller function are pushed on to the stack with the address of instruction just next to the call instruction, this is done so that after the execution of called function, the compiler can return back from where it was sent to the called function.

GATEWAY CLASSES

```
void function1 (int a, int b)
{
    ⋮
    function1 (a, b);
}
```

Direct

Indirection

```
ABC()
{
    xyz()
    ⋮
    xyz()
    {
    ⋮
    ABC()
    }
}
```

GATEWAY CLASSES

2. **Reversing a String:-** It is the characteristics of stack to reverse the order of the execution. This concept can be applied to reverse a string.

3. **Calculation of postfix expression:-**

- Postfix notation is best suitable for a computer to calculate any expression and is universally accepted notation for designing ALU. So any expression is first converted to the postfix notation, stored in stack and then evaluated.
- BODMAS rules, precedence rules etc. are required to evaluate the infix notation which is well acceptable by the human beings.
- It is easy to evaluate the postfix notation because in a postfix expression operands appears before the operators, there is no need for operator precedence or other rules for its evaluation.

GATEWAY CLASSES

Evaluation of Arithmetic Expression

At the time of evaluation of arithmetic expression some precedence rules are to be kept in mind.

(i) Highest Precedence – Exponent Operator ↑ ^ Exp

(ii) Next Highest Precedence – Multiplication and Division Operators * /

(iii) Lowest Precedence – Addition and Subtraction Operators + –

Example.1 :- Evaluate 2 ^ 3 + 4 ^ 2 – 12 / 6. (Ans=22)

2 ↑ 3

2 ^ 3

2 exp 3

2^{2^3}

2^3

Solution:

2 ^ 3 + 4 ^ 2 – 12 / 6

8 + 4 ^ 2 – 12 / 6

8 + 16 – 12 / 6

8 + 16 – 2

24 – 2 = 22

GATEWAY CLASSES

$$\underbrace{2 * 3}_{} \mid 2$$

$$6 \mid 2$$

$$= 3$$

$$\underbrace{2 + 2}_{} - 4$$

$$4 - 4$$

$$= 0$$

GATEWAY CLASSES

Example.2 :- Evaluate each of the following parenthesis free arithmetic expressions.

(a) 5 + 3 ^ 2 – 8 / 4 * 3 + 6 (ans = 14)

(b) 6 + 2 ^ 3 + 9 / 3 – 4 * 5 (ans = -3)

Solution -

(a) $5 + \underbrace{3 \wedge 2} - 8 / 4 * 3 + 6$

$5 + 9 - \underbrace{8 / 4} * 3 + 6$

$5 + 9 - \underbrace{2 * 3} + 6$

$\underbrace{5 + 9} - 6 + 6$

$14 - 6 + 6$

$8 + 6$

$= 14$

GATEWAY CLASSES

(b) $6 + \underbrace{2 \wedge 3} + 9 / 3 - 4 * 5$

$6 + 8 + \underbrace{9 / 3} - 4 * 5$

$6 + 8 + 3 - \underbrace{4 * 5}$

$\underbrace{6 + 8} + 3 - 20$

$14 + 3 - 20$

$17 - 20$

$= -3$

GATEWAY CLASSES

POLISH AND REVERSE POLISH NOTATIONS

Any expression can be represented in the following 3 notations:-

1. A + B – Infix Notation
2. + A B – Prefix Notation (Polish Notation)
3. A B + – Postfix notation (Reverse Polish Notation)

Notation

A + B = Infix

+ A B = Prefix | POLISH NOTATION

A B + = Postfix (REVERSE POLISH NOTATION)

Example 3:- Convert infix (A+B) * C to prefix.
(ans= *+ABC)

Solution -

(A + B) * C

+ A B * C

* + A B C

Ans

GATEWAY CLASSES

Example. 4:- Convert infix A + (B * C) to prefix.

Solution –

$A + (B * C)$

$A + *BC$

$+A*BC$

Ans

Example. 5:- Convert infix (A + B) / (C - D) to prefix.

Solution –

$(A+B) / (C-D)$

$[+AB] / [-CD]$

$/ +AB - CD$ Ans

Sol

$+AB = X$

$-CD = Y$

Pre X / Y

$/ X Y$

$/ +AB \; CD$

GATEWAY CLASSES

Example. 6:- Give postfix for A + [(B + C) + (D + E) * F] / G.

(ANS= ABC+DE+F*+G/+)

Sol –

A + [(B + C) + (D + E) * F] / G.

A + [BC+ + DE+ * F] / G

A + [BC+ + DE+F *] / G

A + BC+DE+F*+ / G

① ② ③

A + BC+DE+F*+G /

ABC+DE+F*+G/+ Ans

Example.7:- Give postfix for A + B - C.

Sol

A + B - C

AB+ - C

-B+C - Ans

GATEWAY CLASSES

Example. 8:- Convert the expression A * B + C / D to postfix form.

Sol.

A * B + C / D

AB * + C / D

AB * + CD /

AB * CD / +

Ans

Example. 9:- Give postfix for A * B + C.

Sol

A * B + C

AB * + C

AB * C +

Ans

GATEWAY CLASSES

Example. 10:- Give postfix form for A + B / C - D.

Sol

$A + B / C - D$

$A + BC/ - D$

$ABC/+ - D$

$ABC/+D-$ Ans

Example. 11:- Give postfix for (A + B) / (C – D).

Sol

$(A + B) / (C - D)$

$[AB+] / [CD-]$

$AB+CD-/$ Ans

GATEWAY CLASSES

Example. 12:- Give postfix form for (A + B) * C / D + E ^ F / G.
(Answer: - A B + C * D / E F ^ G / +)

Sol

$(A+B) * C/D + E \wedge F / G$

$\underbrace{AB+} * C/D + \underbrace{E \wedge F} / G$

$\underbrace{AB+} * \underbrace{C}/D + \underbrace{EF\wedge} / G$

$\underbrace{AB+C*} / \underbrace{D} + \underbrace{EF\wedge} / G$

$\underbrace{AB+C*D/}_{x} + \underbrace{EF\wedge}_{y} / \underbrace{G}_{z}$

$\underbrace{AB+C*D/} + \underbrace{EF\wedge G/}$

$AB+C*D/EF\wedge G/+$ Ans

GATEWAY CLASSES

Example.13:- Give postfix form for A + [(B + C) + (D + E) * F] / G.
(Answer: - A B C + D E + F * + G / +)

Solution:-

$$A + [(B+C) + (D+E) * F] / G$$

$$A + [\underbrace{BC+} + \underbrace{DE+} * \underbrace{F}] / G$$

$$A + [\underbrace{BC+} + \underbrace{DE+F*}] / G$$

$$A + \underbrace{BC+DE+F*+} / G$$

$$\underbrace{A} + \underbrace{BC+DE+F*+G/}$$

$$ABC+DE+F*+G/+$$

Ans

GATEWAY CLASSES

Example.14:- Give postfix form for A + (B * C - (D / E ^ F) * G) * H.
(Answer: - A B C * D E F ^ / G * - H * +)

Sol.

A + (B * C - (D / E ^ F) * G) * H

A + (B * C - (D / EF^) * G) * H

A + (BC* - DEF^/ * G) * H

A + (BC* - DEF^/ * G) * H

A + (BC* - DEF^/G*) * H

A + BC*DEF^/G*- * H

A + BC*DEF^/G*- H *

A BC*DEF^/G*-H*+ Ans

GATEWAY CLASSES

Example. 15:- Convert infix A + (B * C) to prefix.

Sol

A + (B * C)

A + * B C

+ A * B C Ans

Example.16:- Convert infix (A + B) * / (C – D) to prefix.

Soln

(A + B) / (C – D)

[+ A B] / (– C D)

Ans / + A B – C D

GATEWAY CLASSES

AKTU PYQs

Q.1 Give some applications of stack. (AKTU 2019-20, 2 Marks)

Q.2. Give postfix form for A + (B * C - (D / E ^ F) * G) * H. (AKTU 2014-15, 2 Marks)

GATEWAY CLASSES

Download Gateway Classes Application
From Google Play store
Link in Description

ALGO-1 ALGO-2

Next Lecture:- Evaluation of Postfix Expression, Transforming Infix into Postfix, Expression, Reverse of a String using Stack Data Structure PROG

Thank You

GATEWAY CLASSES

B.Tech Third Sem

Data Structure

Unit-2: Lec-3

CS / IT / CS Allied

By- Dr. Kapil Kumar
M.Tech.(CSE), Ph.D.(CSE)

GATEWAY CLASSES

Today's Target

- Evaluation of Postfix Expression ALGO
- Transforming Infix into Postfix Expression
- Reverse of a String using Stack Data Structure
- AKTU PYQs

Reverse Polish Notation

Notation
1. Infix
2. Prefix
3. Postfix

GATEWAY CLASSES

Evaluation of Postfix Expression

Suppose P is an arithmetic expression written in postfix notation. The following algorithm which uses a STACK to hold operands, evaluates P.

ALGORITHM : This algorithm finds the value of an arithmetic expression E written in postfix notation.

Step 1 : Add a right parenthesis ")" at the end of E

Step 2 : Scan E from left to right and repeat steps 3 and 4 for each element of E until the right parenthesis ")" is encountered.

Step 3 : If an operand is encountered, put it on to the STACK.

Step 4 : If an operator * is encountered, then

(a) Remove the top two elements of STACK, where A is the top element and B is the next to top element.

(b) Perform B*A.

(c) Place the result of (b) back on to the STACK. [End of if structure]

[End of step 2 loop]

Step 5 : Set value equal to the top element on STACK.

Step 6 : Exit.

GATEWAY CLASSES

Example.1:- Consider the following arithmetic expression E written in postfix notation:-

E: 5 , 2 , 3 , + , * , 12 , 6 , / , -

Evaluate E- (i) Using Tabular Method. (ii) Using individual stacks for each step. (ans=23)
(iii) Convert this postfix expression into its equivalent infix using tabular method.
(Ans=5*(2+3)-12/6)

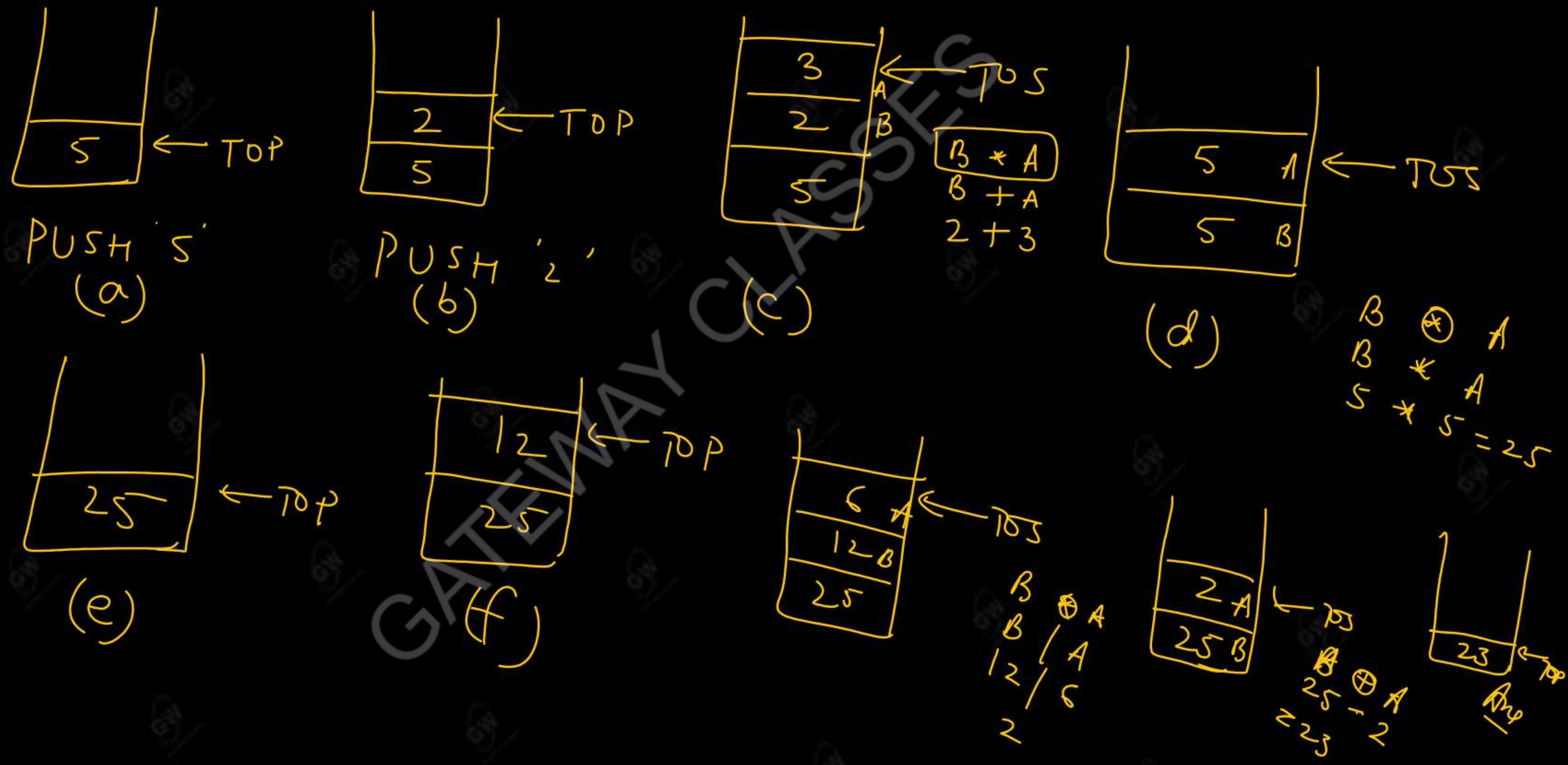
Solution:-

SYMBOL SCANNED(E)	STACK
5	5
2	5, 2
3	5, 2, 3
+	5, 5
*	25
12	25, 12
6	25, 12, 6
/	25, 2
-	23
)	

GATEWAY CLASSES

(II) Through individual stacks for each STACK:

E: 5 , 2 , 3 , + , * , 12 , 6 , / , -



GATEWAY CLASSES

(iii) Conversion of this postfix expression to infix form using tabular method:-

E: 5 , 2 , 3 , + , * , 12 , 6 , / , -

SYMBOL SCANNED (E)	STACK
5	5
2	5, 2
3	5, 2, 3
+	5, (2+3)
*	5 * (2+3)
12	5 * (2+3), 12
6	5 * (2+3), 12, 6
/	5 * (2+3), (12/6)
-	5 * (2+3) - (12/6)
	Ans

GATEWAY CLASSES

$$5 * (2 + 3) - (12 / 6)$$

$$\underbrace{5 * 5} - 2$$

$$25 - 2$$

$$23$$

Ans

GATEWAY CLASSES

Transforming Infix into Postfix Expression

(Reverse Polish Notation)

The following algorithm transform the infix expression Q into its equivalent POSTFIX expression P. This algorithm uses a STACK to temporarily hold operators and left parenthesis. The postfix expressions, P will be constructed from left to right using the operands from Q and the operators which are removed from STACK. We begin by pushing a left parenthesis onto the STACK and adding a right parenthesis at the end of Q. The algorithm is completed when the STACK empty.

Q → P

Infix → Postfix

GATEWAY CLASSES

ALGORITHM : POLISH (Q, P) Suppose Q is an arithmetic expression written in infix notation. This algorithm find the equivalent postfix expression P.

Q infix
P postfix

STEP 1 : PUSH "(" onto STACK, and add ")" to end of Q.

STEP 2 : Scan Q from left to right and repeat steps 3 to 6 for each element of Q until the stack is empty.

STEP 3 : If an operand is encountered, add it to P .

STEP 4 : If a left parenthesis is encountered, push it onto STACK.

STEP 5 : If an operator * is encountered then-

(a) Repeatedly POP from STACK and add to P each operator (on the top of STACK) which has the same precedence as or higher precedence than *.

(b) Add * to stack. [End of if structure]

STEP 6 : If a right parenthesis is encountered then,

(a) Repeatedly POP from STACK and add to P each operator (on the top of stack) until a left parenthesis is encountered.

(b) Remove the left parenthesis (do not add the left parenthesis to P)

[End of if structure] [End of step 2 loop]

STEP 7 : Exit

GATEWAY CLASSES

Example.2:- Translate, each infix expression into its equivalent postfix expression:

(a) (A – B) * (D / E) (ans= AB-DE/*)

Solution:-

Symbol Scanned	Stack	Postfix(P)
(	(	–
A	(	A
–	(–	A
B	(–	A B
)		A B –
*	*	A B –
(	*(	A B –
D	*(	A B – D
/	*(/	A B – D
E	*(/	A B – D E
)	*	A B – D E /
)	–	A B – D E / *

Ans

GATEWAY CLASSES

$$(A - B) * (D / E)$$

$$[AB-] * [DE/]$$

$$AB-DE/*$$

Ans

GATEWAY CLASSES

(b) (A + B ^ D) / (E – F) + G (ANS= ABD^+EF-/G+)

Infix → P.

Solution:-

Symbol Scanned (Q)	Stack	Postfix(P)
(	((	–
A	((	A
+	((+	A
B	((+	AB
^	((+ ^	AB
D	((+ ^	ABD
)	(	ABD^+
/	(/	ABD^+
(	(/(	ABD^+
E	(/(	ABD^+E

GATEWAY CLASSES

Symbol Scanned	Stack	Postfix(P)
—	(/ (-	A B D ^ + E
F	(/ (-	A B D ^ + E F
)	(/	A B D ^ + E F —
+	(+	A B D ^ + E F — /
G	(+	A B D ^ + E F — / G
)	— EMPTY	A B D ^ + E F — / G +

Ans ✓

GATEWAY CLASSES

$$(A + B \wedge D) / (E - F) + G$$

$$(A + [BD\wedge]) / (E - F) + G$$

$$[ABD\wedge +] / [EF-] + G$$

$$[ABD\wedge + EF - /] + G$$

$$ABD\wedge + EF - / G +$$

Ans

GATEWAY CLASSES

(c) A * (B + D) / E – F * (G + H / K) (ANS= ABD+*E/FGHK/+*-)

Solution:-

Symbol Scanned	Stack	Postfix(P)
A	(	A
*	(*	A
(	(* (	A
B	(* (	AB
+	(* (+	AB
D	(* (+	ABD
)	(*	ABD+
/	(/	ABD+*
E	(/	ABD+*E
–	(–	ABD+*E/

GATEWAY CLASSES

Solution:-

Symbol Scanned	Stack	Postfix(P)
F	(−	A B D + * E / F
*	(− *	A B D + * E / F
(	(− * (	A B D + * E / F
G	(− * (	A B D + * E / F G
+	(− * (+	A B D + * E / F G
H	(− * (+	A B D + * E / F G H
/	(− * (+ /	A B D + * E / F G H
K	(− * (+ /	A B D + * E / F G H K
)	(− *	A B D + * E / F G H K / +
)	EMPTY	A B D + * E / F G H K / + * −

Ans

Ans

GATEWAY CLASSES

$$A * (\underbrace{B+D}) / E - F * (G + H / K)$$

$$A * [BD+] / E - F * (G + [HK/])$$

$$A * [BD+] / E - F * [GHK/+]$$

$$[ABD+*] / E - F * [GHK/+]$$

$$[ABD+*E/] - F * [GHK/+]$$

$$[ABD+*E/] - [FGHK/+*]$$

$$\underline{ABD+*E/FGHK/+*-}$$

GATEWAY CLASSES

Example.3 :- Consider the following infix expression Q: ((A + B) * D) ^ (E – F)
Using tabulation method find the equivalent postfix expression P. (ans=AB+D*EF-^)

Solution:-

~~P~~ Q	STACK	Equivalent ~~Infix~~ Postfix Notation (P)
(	((	—
(	(((	—
A	(((	A
+	(((+	A
B	(((+	AB
)	((	AB+
*	((*	AB+
D	((*	AB+D
)	(	AB+D*
^	(^	AB+D*

GATEWAY CLASSES

~~P~~ Q	STACK	~~Equivalent Infix Notation~~ P
(	(^ (	A B + D *
E	(^ (	A B + D * E
−	(^ (−	A B + D * E
F	(^ (−	A B + D * E F
)	(^	A B + D * E F −
)	EMPTY	A B + D * E F − ^

Ans

GATEWAY CLASSES

$$((A + B) * D) \wedge (E - F)$$

$$([AB+] * D) \wedge (E - F)$$

$$[AB+D*] \wedge [EF-]$$

$$AB+D*EF-\wedge$$

Ans

GATEWAY CLASSES

Example.4:- Consider the following arithmetic expression P, written in postfix notation:-

P : 12 , 7 , 3 , - , / , 2 , 1 , 5 , + , * , +

(a) Translate P, by tabular method, into its equivalent infix expression.(ans=12/(7-3)+2*(1+5))

(b) Evaluate the infix expression. (ans=15)

Solution: - (a)

P	Equivalent Infix Notation	
12	12	12
7	12, 7	12, 7
3	12, 7, 3 B ⊕ A	12, 7, 3
−	12, 4 (B, A)	12, (7−3)
/	3	12 / (7−3)
2	3, 2	12 / (7−3), 2
1	3, 2, 1	12 / (7−3), 2, 1
5	3, 2, 1, 5	12 / (7−3), 2, 1, 5
+	3, 2, 6	12 / (7−3), 2, (1+5)
*	3, 12	12 / (7−3), 2 * (1+5)
+	15	12 / (7−3) + 2 * (1+5)

GATEWAY CLASSES

$12 / \underbrace{(7-3)} + 2 * (1+5)$

$12 / 4 + 2 * 6$

$3 + \underbrace{2 * 6}$

$3 + 12$

15

Ans

GATEWAY CLASSES

Solution: - (b) Evaluation of infix expression.

P	Equivalent Infix Notation

GATEWAY CLASSES

Reverse of a String Using Stack Data Structure

```
#include<stdio.h>
#define MAX 20
int top=-1;
char stack[MAX];
int push(char);
char pop();

void main()
{
char str[20];
int i;
clrscr();
printf("\n enter a string:");
gets(str);
for(i=0;i<strlen(str);i++)
push(str[i]);
for(i=0;i<strlen(str);i++)
str[i]=pop();
printf("\n reversed string is:");
puts(str);
getch();
}

push(char item)
{
if(top==MAX-1)
printf("\n Stack Overflow");
else
{
top++;
stack[top]=item;
}
}

char pop()
{
if(top==-1)
printf("\n stack Underflow");
else
return stack[top--];
}
```

RP

Kamal

/* func^n call */

LIFO

/* */ → multiline

→ Single line

'l'
'a'
'm'
'a'
'K'

top

str[0] = l

str[1] = a

GATEWAY CLASSES

AKTU PYQs

Q. 1 Convert the given expression to postfix expression using Stack and show the details of Stack at each step of conversion.

Expression : (a + b * c ^ d) * (e + f / g)

Note: ^ indicates exponent operator. (AKTU 2014-15, 5 Marks)

(ANS = a b c d ^ * + e f g / + *)

Q.2 Convert the following arithmetic infix expression into its equivalent postfix expression.

Expression : A – B / C + D * E + F (AKTU 2017-18, 2 Marks)

(Ans = A B C / - D E * + F +)

Postfix

Q.3 Consider the following infix expression and convert into reverse polish notation using stack:

A + (B * C - (D / E ^ F) * H) (AKTU 2018-19, 7 Marks)

(ANS = A B C * D E F ^ / H * - +)

GATEWAY CLASSES

$(a+b*c\wedge d)*(e+f/g)$

(

(
a
+
b
*
c
)
^
d
)
*

GATEWAY CLASSES

Q.4 Write algorithms for Push and Pop operations in stack. Transform the following expression into its equivalent postfix expression using stack:

A + (B * C - (D / E ^ F) * G) * H

(AKTU 2019-20, 10 Marks)

(Ans= A B D * D E F ^ / G * - H * +)

Q.5 Convert the following infix expression to reverse polish notation expression using stack.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{(2a)}$$

(AKTU 2020-21, 10 Marks)

(Ans= x / b – b 2 ^ 4 a * c * - 1/2 ^ + 2 a * / =)

GATEWAY CLASSES

Q.6 Convert the infix expression (A+B) * (C-D) $ E*F to postfix. Give the answer without any spaces. (AKTU 2021-22, 2 Marks)

(ans= A B + C D – E $ * F *)

Q.7 Write an algorithm for converting infix expression into postfix expression. Trace your algorithm for infix expression Q into its equivalent postfix expression P,

Q : A + (B * c – (D / E ^ F) * G) * H (AKTU 2022-23,10 Marks)

Q.8 Evaluate the following postfix expression using stack.

2 3 9 * + 2 3 ^ - 6 2 / +, Show the contents of each and every steps. Also find the equivalent prefix form of above expression. Where ^ is an exponent operator. (AKTU 2020-21, 10 Marks)

(Ans= 24, infix= 2 + (3 * 9) - (2 ^ 3) + (6 / 2), prefix=+ - + 2 * 3 9 ^ 2 3 / 6 2)

GATEWAY CLASSES

Download Gateway Classes Application
From Google Play store
Link in Description

UNIT 2 –
① Stack
② Queue
→ Ordi

Next Lecture:- Queue Data Structure, Representation of Queue, Insertion and Deletion in Ordinary Queue

Thank You

GATEWAY CLASSES

UNIT-2

33%

- Stack
- Queue
  - ordinary
  - Circular Queue
  - Dequeue
  - Priority
- Recursion & iteration

GATEWAY CLASSES

B.Tech Third Sem

Data Structure

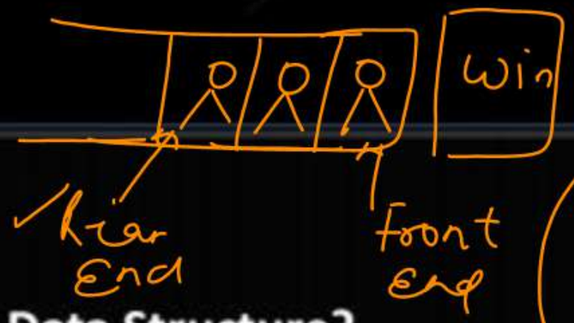
Unit-2: Lec-4

CS / IT / CS Allied

By- Dr. Kapil Kumar
M.Tech.(CSE), Ph.D.(CSE)

GATEWAY CLASSES

DATA STRUCTURE



Today's Target

- What is Queue Data Structure?
- Representation of Queue Data Structure – Static and Dynamic
- Insertion and Deletion Operations on Queue Data Structure and their Algorithms
- Implementation of Insertion and Deletion Operations on Queue Data Structure in C Programming
- Limitations of Ordinary Queue Data Structure

Array

linked list

Simple

GATEWAY CLASSES

1) Simple | ordinary | Queue data structure

2) Circular Queue

3) Double Ended Queue

4) Priority Queue

GATEWAY CLASSES

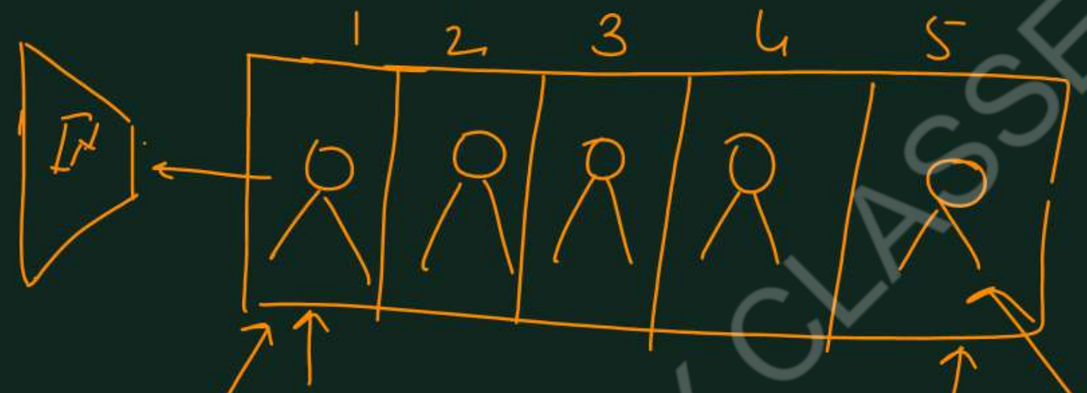
What is Queue Data Structure....

- Queue is also a linear data structure in which operations of Insertion and Deletions are possible as possible with Stack Data Structure but in some different manner.
- Queue data structure is used to solve many problems of computer science. The most common occurrence of a queue data structure in computer science is "job scheduling".
- Queue is also known as FIFO list.
- We have seen that in stack data structure both operations Push and Pop are performed only at one end of data structure and that end is known are Top of the Stack.
- On the other hand, in queue data structure operations are performed on different ends of the data structure such as the operation of insertion of an element is performed from the "Rear" end of the queue and the operation of deletion of an element is performed from the "Front" end of the queue.
- The operation of inserting an element to the queue is sometimes called as Enqueue operations and the operation of deleting the element from the queue is known as Dequeue operation.

GATEWAY CLASSES

FIFO

5, memory cells

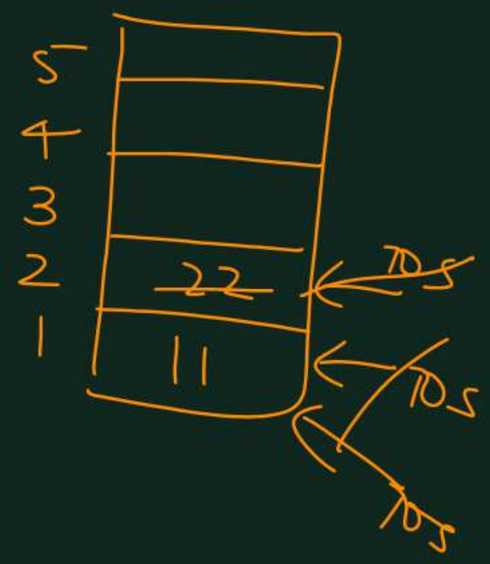


Queue full

Front (Deletion)

Rear (Insertion)

Electricity Bill



GATEWAY CLASSES

Example.1:- Consider the following empty queue, with 5 memory cells as:-

Queue:

1	2	3	4	5

Initially Empty (front = Rear = 0)

As the queue is empty Front and Rear end both are at 0.

Queue

0	1	2	3	4

Initially empty Array

f = R = -1

(i) Insert A -

1	2	3	4	5
A				

Rear

front

Insert A

Rear = Rear + 1
= 0 + 1
= 1

Queue[Rear] = A

If (front = 0)
set front = 1

~~(ii) Insert B -~~

~~(iii) Insert C -~~

GATEWAY CLASSES

(ii) Insert B

Queue:

1	2	3	4	5
A	B			

FRONT ~~REAR~~ R

Rear = Rear + 1
= 1 + 1 = 2

Queue[Rear] = B
if (front == 0) F

GATEWAY CLASSES

(iii) Insert C

queue:

1	2	3	4	5
A	B	C		
F	~~R~~	R		

$R = R + 1 = 2 + 1$

$= 3$

$Queue[R] = C$

GATEWAY CLASSES

(iv) Insert D –

Queue

1	2	3	4	5
A	B	C	D	

F, ~~R~~, R

$R = R + 1 = 3 + 1 = 4$

$Queue[R] = D$

(v) Insert E -

Queue

1	2	3	4	5
A	B	C	D	E

F, ~~R~~, R ← maxsize

$R = R + 1 = 4 + 1 = 5$

$Queue[R] = E$

Queue full — if (F == 1) && (R == maxsize)

Now, the queue is full because the rear end is equal to the maximum size of the array.

~~**(vi) Delete A -**~~

in 'C' Program

Queue full

if((F == 0) && (R == maxsize-1))

0	1	2	3	4
A	B	C	D	E

F, R

~~**(vii) Delete B -**~~

GATEWAY CLASSES

(VI) Delete A

Queue :

1	2	3	4	5
A	B	C	D	E

~~R~~ F R

item = Queue[f]

item = A

f = f + 1

GATEWAY CLASSES

Delete B

	1	2	3	4	5
Queue		~~B~~	C	D	E
		~~F~~	F		R

item = Queue[f]
item = B
f = f + 1

{ int a = 2;
b = a;

GATEWAYCLASSES

Algorithm : Queue Insertion or Enqueue

This algorithm is used to insert an element ITEM to the Rear end of the Queue which is of n elements (maximum size of Queue).

Step 1:- [Check for the Overflow]

If (Rear == n) then

if (Rear == maxsize)

Print " Overflow" or "Insertion operation is not possible as Queue is full".

[End of if structure]

if (Rear == maxsize-1)

OVERFLOW

UNDERFLOW

Step 2:- Set: Rear = Rear + 1

Step 3:- Set: Queue [Rear] = ITEM.

Step 4:- if (Front == 0) then

Set : Front = 1. {if the first ITEM is inserted to the Queue}

[End of if structure]

Step 5:- Exit

GATEWAY CLASSES

Algorithm : Queue Deletion or Dequeue

This algorithm is used to delete an element ITEM from the Front end of the Queue which is of n elements (i.e. maximum size of Queue).

Step 1:- [Check for Underflow]

If (Front == 0) then

Print " Underflow" or "Deletion operation is not possible as queue is empty".

[End of if structure]

Step 2:- ITEM = Queue [Front]

Step 3:- if (Front == Rear) then

{

Set: Front = 0.

Set: Rear = 0.

}

else

{

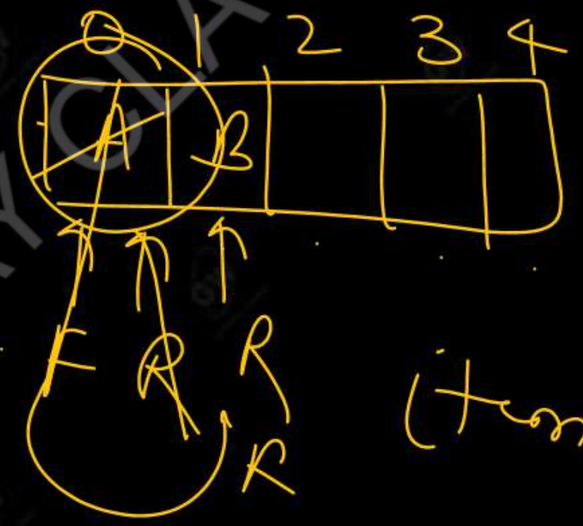
Set : Front = Front + 1.

}

Step 5:- Exit

if (f == -1)

Queue:



item = Queue[f]

GATEWAY CLASSES

Implementation of Insertion & Deletion Operations of Queue Data Structure in C

```
/*Insertion and Deletion operations to the Ordinary Queue*/
#include<stdio.h>
void queue_insertion(int);
void queue_deletion();
void traverse();
int queue[20];
int front=0,rear=0,item;
int size;
void main()
{
int ch;
clrscr();
printf("\n Enter the size of the queue:");
scanf("%d",&size);
if(size>20)
{
printf("\n Invalid size of queue, it should be less than 20
because the size of the array is 20");
getch();
exit();
}
do
{
printf("\n Your choices are as follows:");
printf("\n\n Press 1 for Insertion of an element to the Queue");
printf("\n\n Press 2 for Deletion of an element from the
Queue");
printf("\n\n Press 3 for Exit from the Program:");
printf("\n\n Enter your choice:");
scanf("%d",&ch);
```

GATEWAY CLASSES

Implementation of Insertion & Deletion Operations of Queue Data Structure in C

```
switch(ch)
{
case 1:
queue_insertion(size);
printf("\n After insertion the queue as follows");
traverse();
break;
case 2:
queue_deletion();
printf("\n After deletion the queue is as follows:");
traverse();
break;
case 3:
exit();
}
printf("\n Do you want to continue, press 1 for
continue and 0 for end:");
scanf("%d",&ch);
}while(ch!=0);
getch();
}
```

GATEWAY CLASSES

Implementation of Insertion & Deletion Operations of Queue Data Structure in C

```
void queue_insertion(int s)
{
printf("\n Enter the item to be inserted to the queue:");
scanf("%d",&item);
if(rear<s)
{
rear++;
queue[rear]=item;
if(front==0)
front=1;
}
else
printf("\n Insertion is not possible:");
}

void queue_deletion()
{
if(front==0)
{
printf("\n Underflow");
return;
}
else
{
item=queue[front];
printf("\n Deleted item is =%d",item);
}
if(front==rear)
{
front=0;
rear=0;
}
else
front=front+1;
}

void traverse()
{
int i;
if(front==0)
return;
for(i=front;i<=rear;i++)
printf("\n %d",queue[i]);
}
```

3

11, 22, 33

1<3 T

2<3 T

2<2 F

S-1

OVERFLOW

-1 F

0

-1

R

-1

11 22 33

-1 :

-1 :

Queue

0	1	2
11	22	33

F

R

GATEWAY CLASSES

Limitations of Simple or Ordinary Queue

There are some limitations associated with simple or ordinary queue.

Consider the following initially empty queue of 6 memory cells:-

0	1	2	3	4	5
11	22	33	44	55	66

Queue:

full if F = 0 && R = maxsize-1

We analyze the problem associated with simple queue with a series of insertion and deletion operations performed on the simple queue.

For example, now insert 11, 22, 33, 44, 55, 66 in this queue one by one-

Del Insert

GATEWAY CLASSES

Limitations of Ordinary Queue

We know that, in ordinary queue the elements are deleted from the Front end and inserted to the Rear end of the queue.

Suppose one element is deleted from the Front end of the queue when the queue is full as mentioned in the previous example. The deletion operation can be performed like this-

Now, there is one empty memory cell in the queue but it is not possible to insert new element to the queue because it is pre-fixed that insertion of element in the simple queue is possible only from the Rear end of the queue and in this example when the queue is full the Rear is on the last position of the queue that is Rear is equal to the maximum size of the queue because the queue is full.

GATEWAY CLASSES

Limitations of Ordinary Queue

So, space is available but insertion of new element in simple queue is not possible because of its limitation.

If we will try to insert an element in this queue, it will give message " Overflow" which is not correct.

This limitation can be removed by using the next version of queue data structure which is known as Circular Queue.

Circular Queue

GATEWAY CLASSES

Download Gateway Classes Application
From Google Play store
Link in Description

Next Lecture:- Circular Queue Data Structure, Representation of Circular Queue, Insertion and Deletion in Circular Queue with implementation in C programming

Thank You

GATEWAY CLASSES

B.Tech Third Sem

Data Structure

Unit-2: Lec-6

CS / IT / CS Allied

By- Dr. Kapil Kumar
M.Tech.(CSE), Ph.D.(CSE)

GATEWAY CLASSES

Today's Target

⇒ limitations of ordinary/ simple Queue

- What is Circular Queue Data Structure?
- Insertion and Deletion Operations on Circular Queue
- Algorithms of Insertion and Deletion of an item in Circular Queue
- Implementation of Insertion and Deletion Operations on Circular Queue through C Programming
- AKTU PYQs

Queue
- Simple/ordinary
- Circular
- DE Queue
- Priority Queue

GATEWAY CLASSES

UNIT-2

1. Stack
2. Queue
3. Iteration & Recursion

GATEWAY CLASSES

Limitations of Ordinary Queue

Consider the following empty queue of 6 memory cells:-

0	1	2	3	4	5

(i) 1 2 3 4 5 6

Now insert 11, 22, 33, 44, 55, 66 in this queue-

maxsize = 6

Front = (Deletion) = −1
Rear = (Insertion) = −1
} NULL

0	1	2	3	4	5 (maxsize−1)
11	22	33	44	55	66

Front

0	1	2	3	4 Rear	5 maxsize−1
	22	33	44	55	66

F R

GATEWAY CLASSES

0	1	2	3	4	5
		33	44	55	66
	~~F~~	T			R

item = Q[F]

item = 22

f = f + 1

= 1 + 1 = 2

GATEWAY CLASSES

Limitations of Ordinary Queue

- In ordinary queue the elements are deleted from the front end and inserted to the Rear end of the queue.
- Suppose one element is deleted from the Front end of the queue when the queue is full as mentioned above.
- Now, there is one empty memory cell but it is not possible to insert new element to the queue because it is fixed that insertion of element is possible only from the Rear end of the queue and in this example when the queue is full the Rear is on the last position of the queue that is Rear is equal to the maximum size of the queue because the queue is full.

  maxsize-1
- So, space is available but insertion of new element is not possible because of its limitation. This limitation can be overcome by using the next version of queue which is known as Circular Queue.

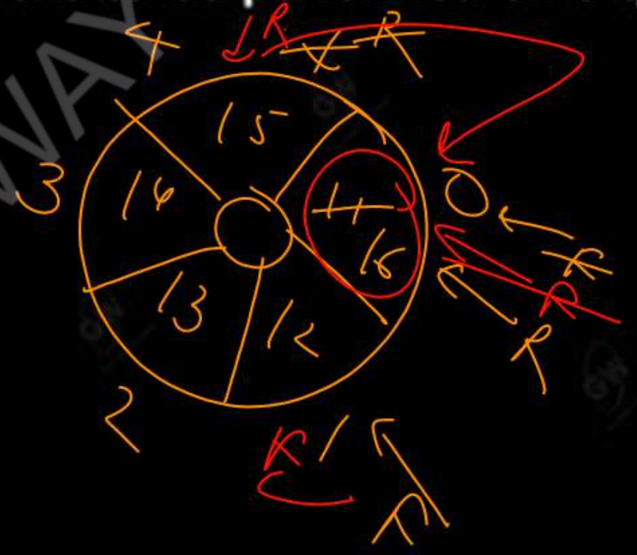
GATEWAY CLASSES

Circular Queue

- Circular queue is form of a queue where there is no end of queue.
- Suppose we have one queue of 5 elements, it means front represent the first element of the queue and rear represent the last element of the queue. In circular queue, rear can move to the first location according the requirement.
- This concept of circular queue can be understood by the following example:-

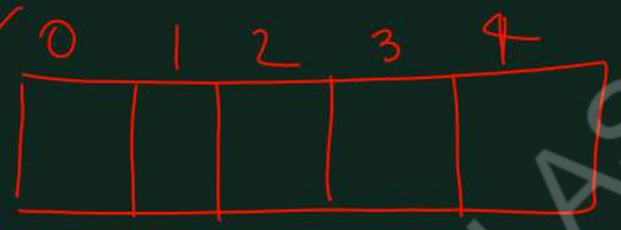
Example 1:- Consider the following operations to be performed on a queue when it is circular with 5 memory cells:-

~~(a) Queue is initially empty-~~

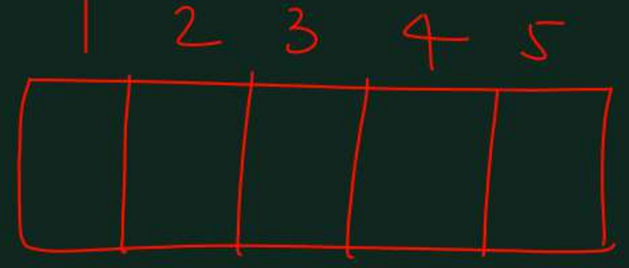


GATEWAY CLASSES

(a) Queue is initially empty :-



Front = -1
Rear = -1



front = 0
Rear = 0

GATEWAY CLASSES

(a)A is inserted to the queue-

Circular Queue :

0	1	2	3	4
A				

Rear
front

(2) Queue[Rear] = 'A'

(1) Rear = Rear+1
= -1+1
= 0

if (front == -1) then
set front = 0

(c) B and C are inserted to the queue-

(i) B

0	1	2	3	4
A	B			

f R R

R = R+1
= 0+1 = 1
Q[R] = B

GATEWAY CLASSES

(ii) C

0	1	2	3	4
A	B	C		
F	~~R~~	R		

$R = R + 1$

$= 1 + 1 = 2$

$Q[R] = C$

GATEWAY CLASSES

(d) D is inserted to the queue-

0	1	2	3	4
A	B	C	D	
F		R	R	

$R = R + 1$

$= 2 + 1 = 3$

$Q[R] = D$

(e) Two items are deleted from the queue-

0	1	2	3	4
~~A~~	~~B~~	C	D	
~~F~~ ~~R~~	~~F~~ ~~R~~	F	R	

GATEWAY CLASSES

(f) E , F and G are inserted to the queue-

0	1	2	3	4
F	G	C	D	E
~~F~~	R	F	~~R~~	~~R~~

(i) E

R = R+1
= 3+1
= 4

Q[R] = E

(ii) F

R = R+1
= 4+1 = 5 X

if (R == maxsize-1)
4 == 5-1
4 == 4 T

&& f != 0
set R = 0
Q[R] = F

(iii) G

R = R+1
Q[R] = G

Simple

(it should be noted here if it is an ordinary queue then insertion of F and G is not possible which is the limitation of ordinary queue

GATEWAY CLASSES

X is inserted to the Queue :- OVERFLOW
{Because f is equal to the R+1}

(g) Two items are deleted from the queue-

0	1	2		4
F	G	~~C~~	~~D~~	E
R	~~F~~	~~F~~	~~F~~	F

if(F == R + 1)

2 == 1 + 1

2 == 2 True

(f) Two items H and I are inserted-

0	1	2	3	4
~~F~~	G	H	I	~~E~~
~~F~~	~~F~~ R	~~F~~ ~~R~~	~~F~~ ~~R~~	~~F~~ ~~R~~

if F == R + 1

4 == 3 + 1

4 == 4 True

GATEWAY CLASSES

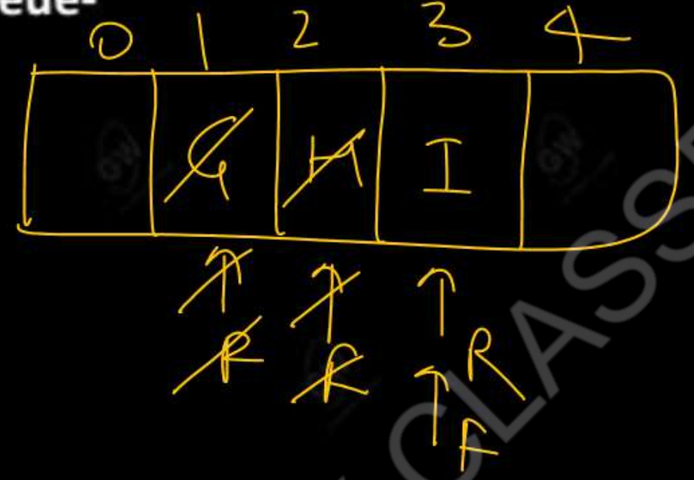
(i) I is inserted to the queue-

Now insertion is not possible because queue is full.

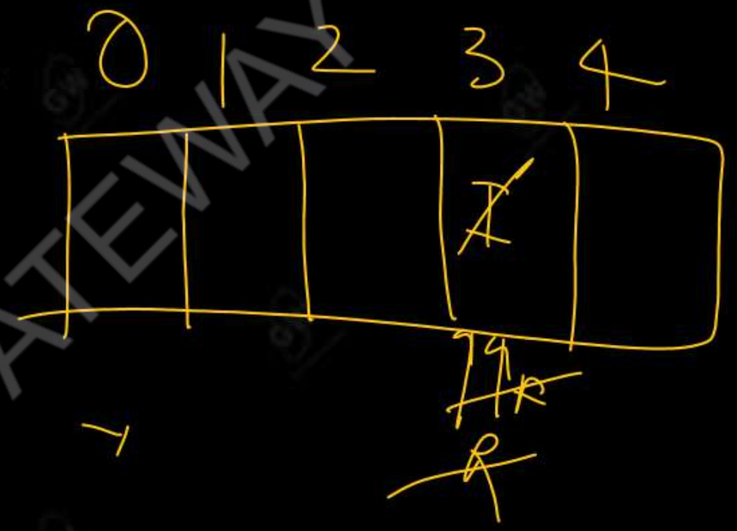
(j) Two items are deleted from the queue-

GATEWAY CLASSES

(k) Two items are deleted from the queue-



(l) One item is deleted from the queue-



item = Q[f]
item = I
if (f == R)
set f = R = -1

GATEWAY CLASSES

(m) One item is deleted from the queue-

Now deletion is not possible because underflow.

GATEWAY CLASSES

Example 2:- Consider the following circular queue capable of accommodating maximum 6 elements:

Rear = 4, Front = 2

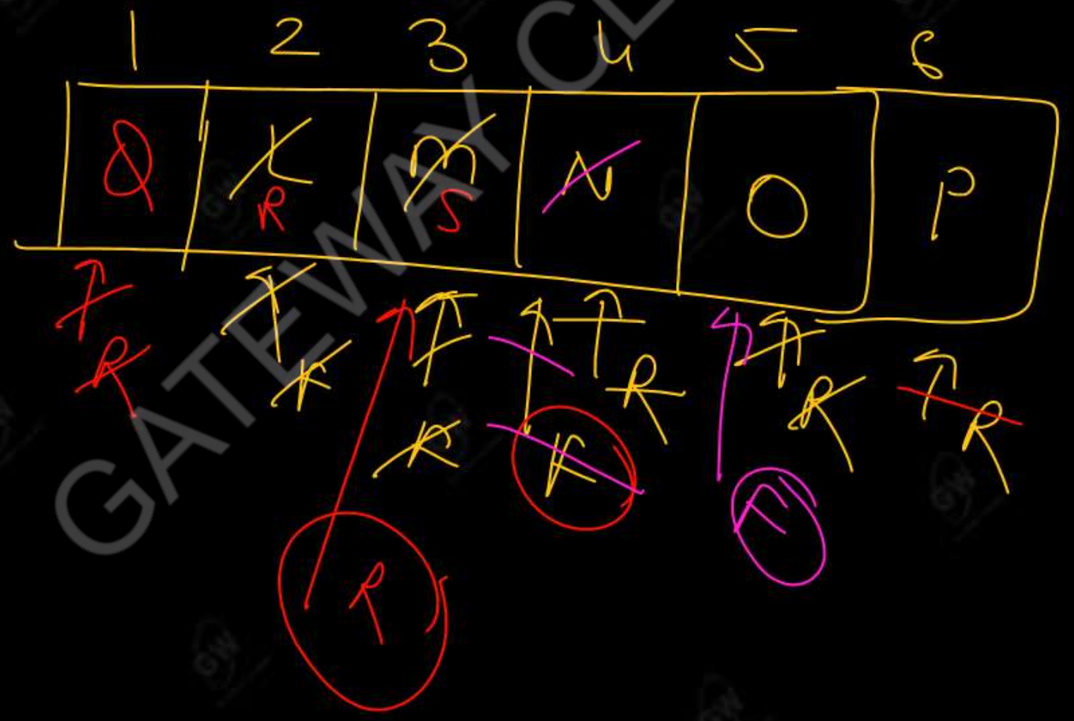
Queue :- _ , L , M , N , _ , _ .

Describe the queue as the following operations take place-

(a) Add O (b) Add P (c) Delete 2 letters (d) Add Q, R, S (e) Delete one letter

Solution:-

Initial Queue:-



(a) Add O

GATEWAY CLASSES

(b) Add P

(c) Delete 2 letters

GATEWAY CLASSES

(d) Add Q, R, S

(e) Delete one letter

GATEWAY CLASSES

Algorithms Circular Queue Insertion and Deletion

Algorithm : Circular Queue Insertion

This algorithm is used to insert an element ITEM to the circular queue.

Most Imp

Step 1:- if (Front == 0 && Rear == MaxSize-1) || (Front == Rear + 1) then

Q

Write "Overflow" or "Insertion of element is not possible as queue is full"

[end of if structure]

Step 2:- if Front == -1 then

Set Front = 0 and Rear = 0

Queue[Rear] = ITEM.

Step 3:- else if Rear == Maxsize-1 then

Set Rear = 0

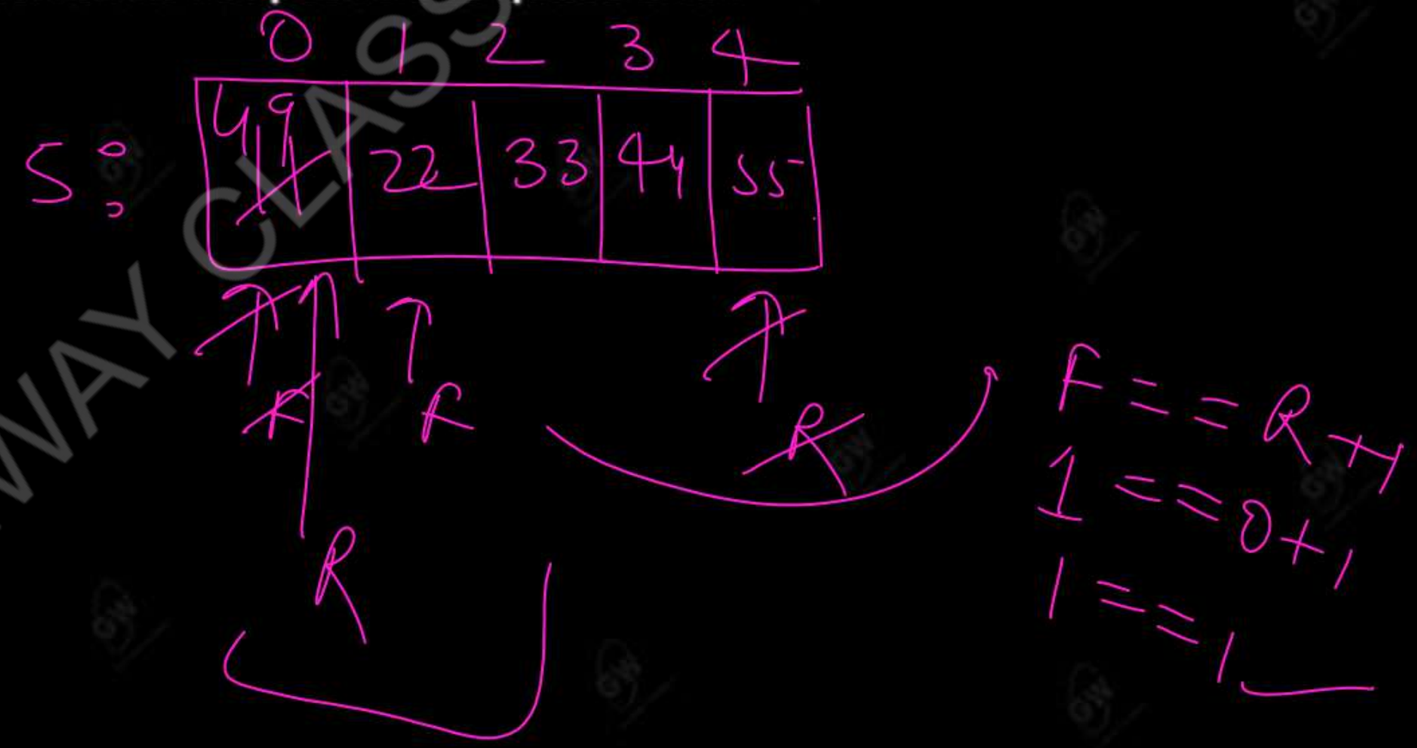
Queue [Rear] = ITEM.

Step 4:- else

Rear = Rear + 1

Queue [Rear] = ITEM.

Step 5:- Exit



GATEWAY CLASSES

```
R = R + 1
Q[R] = item
if (f == -1)
    set f = f + 1
```

GATEWAY CLASSES

Algorithm : Circular Queue Deletion

This algorithm is used to delete an element ITEM from the circular queue.

Step 1:- if Front == - 1 then

Write "Underflow" or "Deletion operation is not possible as queue is empty".

[end of if structure]

Step 2:- [Read ITEM from the Front of the Queue]

ITEM = Queue [Front].

Step 3:- if Front == Rear then

Set Front = -1 and

Set Rear = -1.

Step 4:- else if Front = Maxsize then

Set: Front = 0.

Step 5:- else

Set Front = Front + 1.

Step 6:- Exit

GATEWAY CLASSES

Insertion and Deletion of an element in Circular Queue

```
/*Insertion and Deletion operations to the Circular Queue*/
#include<stdio.h>
#define MAX 5
void insertion();
void deletion();
void display();
int cqueue[MAX];
int front=-1,rear=-1;
void main()
{
int choice;
clrscr();
while(1)
{
printf("\n Your choices are as follows:");
printf("\n\n 1:Insertion");
printf("\n\n 2:Deletion");
printf("\n\n 3:Display");
printf("\n\n 4:Quit");
printf("\n\n Enter your choice:");
scanf("%d",&choice);
switch(choice)
{
case 1:
insertion();
break;
case 2:
deletion();
break;
case 3:
display();
break;
case 4:
exit(1);
default:
printf("\n This choice is not available:");
}
}
getch();
}
```

GATEWAY CLASSES

```
void insertion()
{
int item;
if((front==0 && rear==MAX-1) || (front==rear+1))
{
printf("\n Queue is Overflow");
return;
}
if(front==-1)    /*if queue is empty*/
{
front=0;
rear=0;
}
else if(rear==MAX-1) /*rear is at the last position*/
rear=0;
else
rear=rear+1;
printf("\n Enter the item to be inserted to the queue:");
scanf("%d",&item);
cqueue[rear]=item;
}
```

```
void deletion()
{
if(front==-1)
{
printf("\n Queue Underflow");
return;
}
printf("\n Deleted item from queue is =%d",cqueue[front]);
if(front==rear)
/* Queue has only one element*/
{
front=-1;
rear=-1;
}
else
if(front==MAX-1)
front=0;
else
front=front+1;
}
```

GATEWAY CLASSES

```
void display()
{
int front_pos=front;
int rear_pos=rear;
if(front==-1)
{
printf("\n queue is empty");
return;
}
printf("\n queue elements are as follows:");

if(front_pos<=rear_pos)
while(front_pos<=rear_pos)
{
printf("\n%d",cqueue[front_pos]);
front_pos++;
}
else
{
while(front_pos<=MAX-1)
{
printf("\n%d",cqueue[front_pos]);
front_pos++;
}
front_pos=0;
while(front_pos<=rear_pos)
{
printf("\n%d",cqueue[front_pos]);
front_pos++;
}
}
printf("\n");
}
```

GATEWAY CLASSES

AKTU PYQs

C Prog

Q.1 How would you implement a circular queue in C using array? Write routine to implement operations for it.

(AKTU 2014-15, 10 Marks)

Q.2 Explain circular queue. What is the condition if circular queue is full?

(AKTU 2017-18, 2 Marks)

Q.3 Write the syntax to check whether a given circular queue is full or empty?

(AKTU 2018-19, 2 Marks)

Q.4 Explain how a circular queue can be implemented using arrays. Write all functions for circular queue operations.

(AKTU 2018-19, 7 Marks)

Q.5 Write the full and empty condition for a circular queue data structure.

(AKTU 2020-21, 2 Marks)

Q.6 Write the condition for empty and full circular queue. (AKTU 2022-23, 2 Marks)

Q.7 What is circular queue? Write a C code to insert an element in circular queue? (AKTU 2022-23, 10 Marks)

GATEWAY CLASSES

Download Gateway Classes Application
From Google Play store
Link in Description

Dequeue Deque

Next Lecture:- Double Ended Queue Data Structure, Insertion and Deletion Operations in Double Ended Queue

Thank You

GATEWAY CLASSES

B.Tech Third Sem

Data Structure

Unit-2: Lec-6

CS / IT / CS Allied

By- Dr. Kapil Kumar
M.Tech.(CSE), Ph.D.(CSE)

GATEWAY CLASSES

Today's Target

- What is Double Ended Queue or DEQUE?
- Types of DEQUE : Input Restricted and Output Restricted DEQUE
- Insertion and Deletion Operations in DEQUE
- Algorithms to Insert an item to DEQUE (to the front(left) end and rear (right) end)
- Algorithms to Delete an item from DEQUE (from the front (left) end and from the rear (right) end)
- AKTU PYQs

GATEWAY CLASSES

Double Ended Queue

- In a DEQUE, both insertion and deletion operations can be performed at either end of the DEQUE i.e. any item can be inserted from both front or rear ends and in the same manner the operation of deletion of an element can also be performed from both front and rear ends of the DEQUE.
- Consider the following figure to understand the concept:-

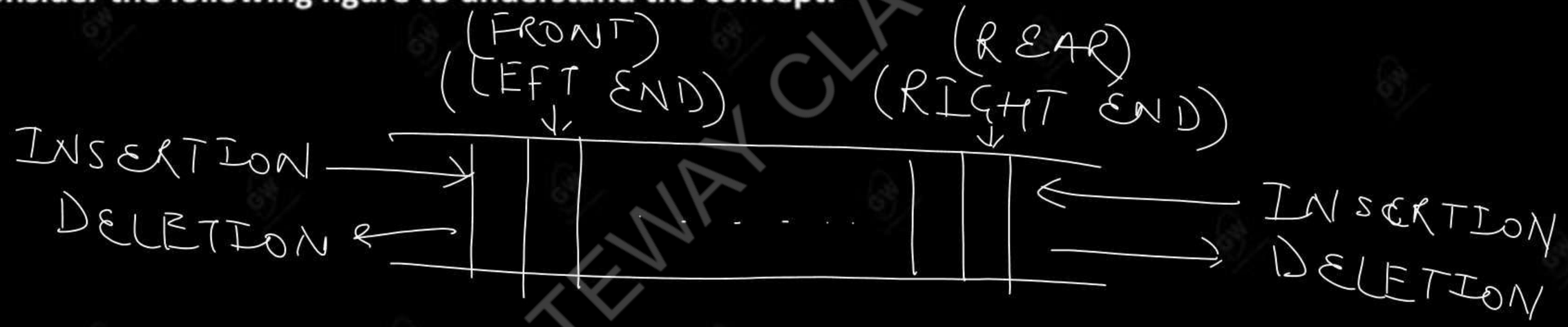


FIG:- A VIEW OF DEQUE

- DEQUE can be represented by using doubly linked list or a circular linked list.

GATEWAY CLASSES

Types of DEQUE

DEQUE can be of 2 types-

1. Input Restricted DEQUE (INPUT / INSERTION = Only one end, Del – Both ends)
2. Output Restricted DEQUE (Output / Deletion = Only one end, Insertion = Both ends)

1. **Input Restricted DEQUE –** In Input – restricted DEQUE, the item can be deleted from both the ends of the DEQUE but insertion can be performed only at one end of the DEQUE.

Consider the following figure to understand the concept-

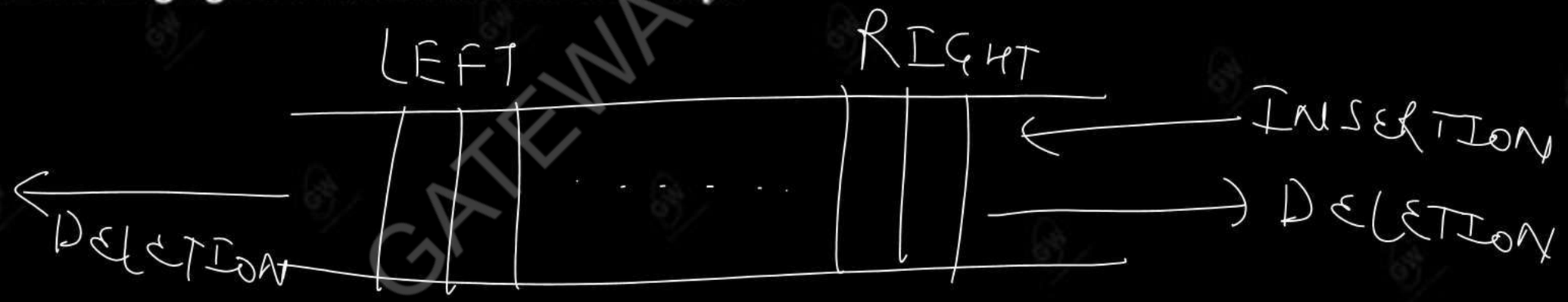


Fig:- Input Restricted DEQUE

GATEWAY CLASSES

2. Output - Restricted DEQUE – In Output – restricted DEQUE, the item can be inserted from both the ends of the DEQUE but operation of deletion can be performed only at one end of the DEQUE.

Consider the following figure to understand the concept-

Fig:- A view of Output Restricted Deque.

GATEWAY CLASSES

Possible Operations on Deque

The possible operations which can be performed on DEQUE are as follow:-

1. Insertion of an element at the rear (right) end of the DEQUE
2. Insertion of element at the front (left) end of the DEQUE
3. Deletion of an element from the rear (right) end of the DEQUE
4. Deletion of an element from the front (left) end of the DEQUE

GATEWAY CLASSES

Example 1:- Consider the following queue of characters where DEQUE is a circular array which is allocated 6 memory cells:

deque

LEFT = 2 RIGHT = 4 DEQUE : _ , A , C , D , _ , _

Describe the DEQUE while the following operations take place-

(a) F is added to the right of the DEQUE

(b) Two letters on the right are deleted

(c) K, L and M are added to the left of the DEQUE

(d) One letter on the left is deleted

(e) R is added to the right of the DEQUE

(f) S is added to the left of the DEQUE

GATEWAY CLASSES

Solution:-

(a) F is added to the right of the DEQUE-

Initial Deque:-

1	2	3	4	5	6
	A	C	D		
	↑ left		↑ Right		

1	2	3	4	5	6
	A	C	D	F	
				↑ Right	

if (left == 0 && right == Max size) || F = R+1 L = Right+1

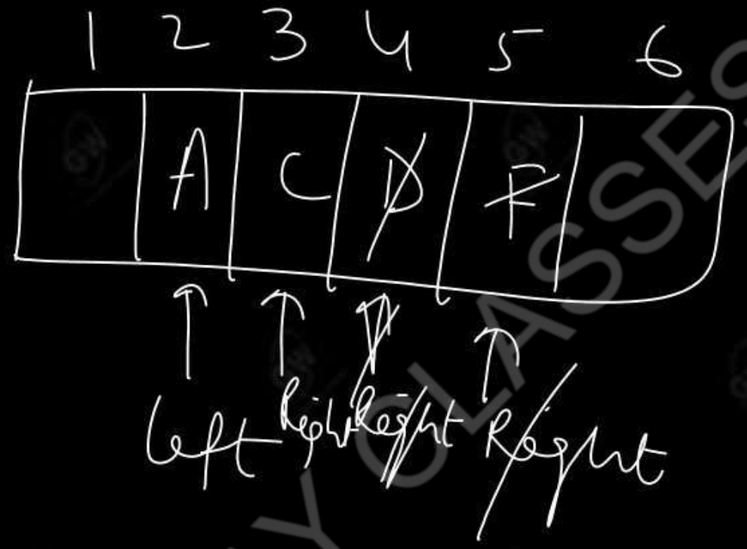
Right = Right + 1
= 4+1 = 5

Deque[R] = item

Deque[5] = F

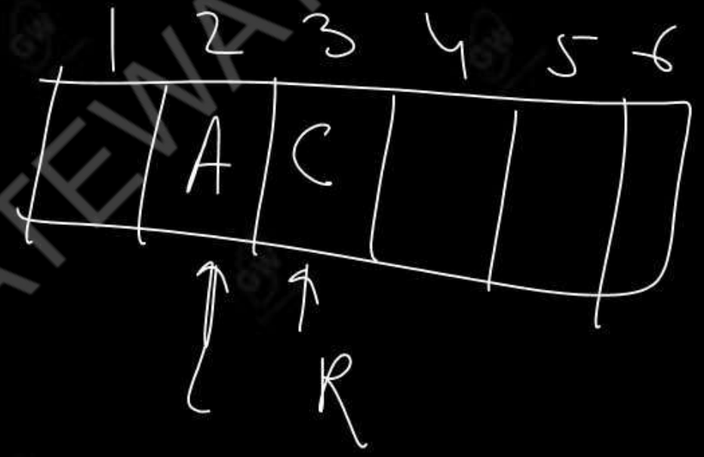
GATEWAY CLASSES

(b) Two letters on the right are deleted



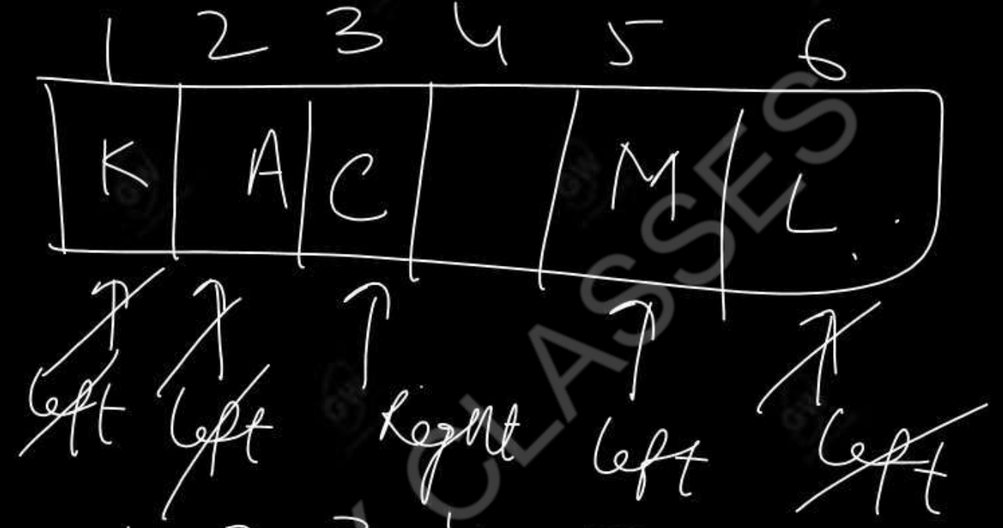
item = Deque[Right]

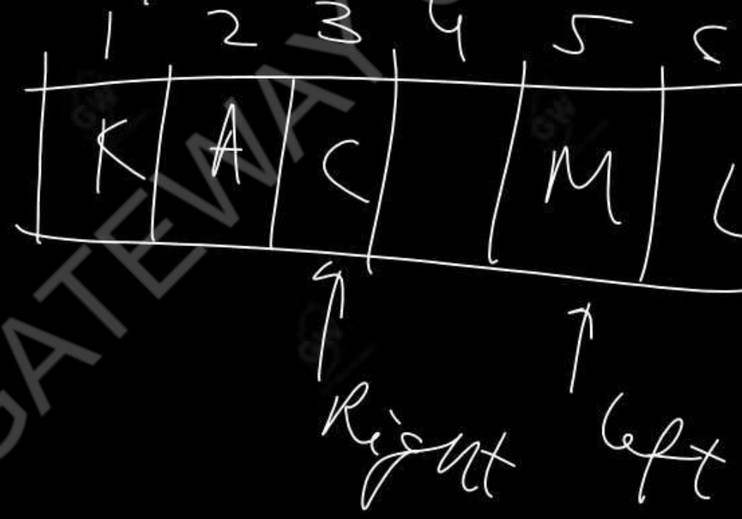
Right = Right - 1



GATEWAY CLASSES

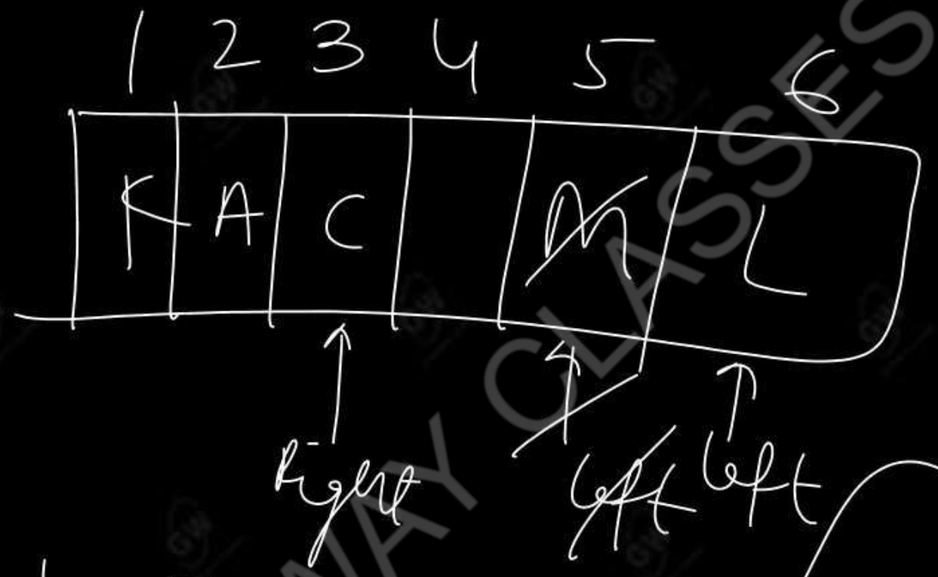
(c) K, L and M are added to the left of the DEQUE

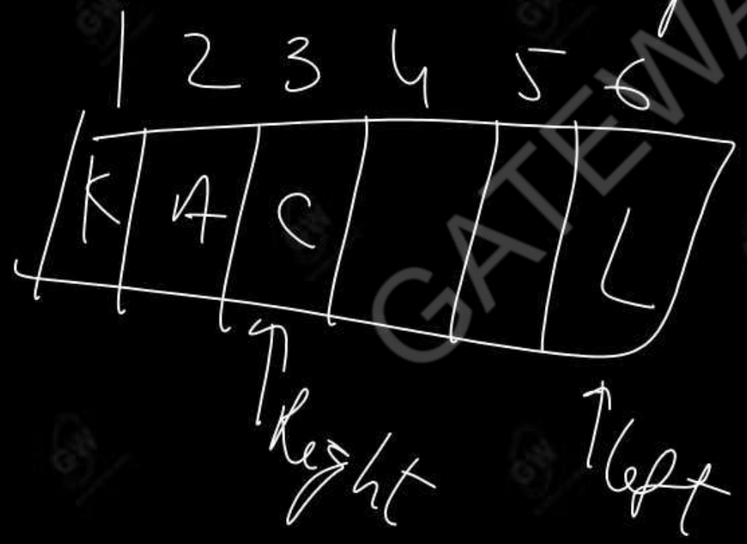




GATEWAY CLASSES

(d) One letter on the left is deleted





data = deque[left]
data = M
left = left + 1

GATEWAY CLASSES

(e) R is added to the right of the DEQUE

1	2	3	4	5	6
K	A	C	R		L

Right

~~Right~~

Left

1	2	3	4	5	6
K	A	C	R		L

Right

Left

Right = Right + 1

= 3 + 1 = 4

Deque[Right] = item

Deque[4] = R

GATEWAY CLASSES

(f) S is added to the left of the DEQUE

1	2	3	4	5	6
K	A	C	R	S	L

Right left ~~left~~

left = Right + 1
5 = 4 + 1
5 = 5 ✓ T

1 | 6

1	2	3	4	5	6
K	A	C	R	S	L

Right left

left = left − 1
= 6 − 1 = 5

Deque[left] = S
Deque[5] = S

GATEWAY CLASSES

Example 2:- Consider the DEQUE maintained by a circular array with N memory cells-

(a) Suppose an element is added to the DEQUE. How is LEFT and RIGHT changed?

(b) Suppose an element is deleted from the DEQUE. How is LEFT and RIGHT changed?

Solution:-

(a) If the element is added on the left, then LEFT decreased by 1 (mod N). On the other hand, if the element is added on the right, then RIGHT is increased by 1.(mod N)

(b) If the element is deleted from the left, the LEFT is increased by 1 (mod N). However if the element is deleted from the right, then RIGHT is decreased by 1 (mod N). In the case that LEFT = RIGHT before the deletion (i.e. when the DEQUE has only one element), then LEFT and RIGHT are both assigned to NULL (or -1) to indicate that the DEQUE is empty after the deletion of only one element of the deque.

GATEWAY CLASSES

Algorithm: Insertion DEQUE Right (rear) Side

This algorithm is used to insert an element at the right side of the DEQUE.

Step 1:- Read "Item" to be inserted to the DEQUE.

Step 2:- [Check for overflow]

If (left==0 && right == maxsize – 1) || (left == right + 1) then

Write " Overflow" or "DEQUE is full"

[End of if structure]

Step 3: - if (left == -1)

Set left = 0.

Set right = 0

Step 4: - else if (right = maxsize -1) then

Set right = 0

else

right = right + 1.

Step 5: - DEQUE[right] = item.

Step 6: - Exit

GATEWAY CLASSES

Algorithm: Insertion DEQUE Left (front) Side

This algorithm is used to insert an element at the left side of the DEQUE.

Step 1:- Read "Item" to be inserted to the DEQUE.

Step 2:- [check for overflow]

If (left==0 && right == maxsize – 1) || (left == right + 1) then

Write " Overflow" or "DEQUE is full"

[End of if structure]

Step 3: - if (left == - 1)

Set left = 0.

Set right = 0

Step 4: - else if (left == 0) then

Set left = maxsize - 1

else

left = left - 1.

Step 5: - DEQUE [left] = item.

Step 6: - Exit

GATEWAY CLASSES

Algorithm: Deletion DEQUE Right (rear) Side

This algorithm is used to delete an element from the right side of the DEQUE.

Step 1:- if left = = -1 then

Write " underflow" or "DEQUE is empty"

[End of if structure]

Step 2:- item = DEQUE [right]. ← Deletion

Step 3: - if (left == right) then

Set left = -1.

Set right = -1

Step 4: - else if (right == 0) then

Set right = maxsize -1

else

right = right - 1.

Step 5: - Exit

GATEWAY CLASSES

Algorithm: Deletion in DEQUE from the Left (front) Side

This algorithm is used to delete an element from the right side of the DEQUE.

Step 1:- if left = = -1 then

Write " underflow" or "DEQUE is empty"

[End of if structure]

Step 2:- item = DEQUE [left].

Step 3: - if (left == right) then

Set left = -1.

Set right = -1

Step 4: - else if (left == maxsize - 1) then

Set left = 0

else

left = left + 1.

Step 5: - Exit

GATEWAY CLASSES

AKTU PYQs

Q.1 Differentiate between DEQUE and priority queue.

(AKTU 2014-15, 5 Marks)

Q.2 Discuss array and linked representation of queue data structure. What is DEQUE?

(AKTU 2019-20, 10 Marks)

GATEWAY CLASSES

Download Gateway Classes Application
From Google Play store
Link in Description

Next Lecture:- Priority Queue, Representation of Priority queue, some major application of Queue, Difference between Stack and Queue

Thank You

GATEWAY CLASSES

UNIT-2

1. Stack
2. Queue
3. Iteration & Recursion

GATEWAY CLASSES

B.Tech Third Sem

Data Structure

Unit-2: Lec-7

CS / IT / CS Allied

By- Dr. Kapil Kumar
M.Tech.(CSE), Ph.D.(CSE)

GATEWAY CLASSES

DATA STRUCTURE

GW GATEWAY CLASSES

Today's Target

- What is Priority Queue Data Structure?
- Types of Priority Queue
- Representation of Priority Queue
- Application of Queues Data structure
- Difference between Stack and Queue Data Structures
- AKTU PYQs

GATEWAY CLASSES

AKTU PYQs

Q.1 Differentiate between deque and priority queue. (AKTU 2014-15, 5 Marks)

Q.2 Write short note on Priority Queue. (AKTU 2016-17, 5 Marks)

Q.3 What do you mean by priority queue? (AKTU 2017-18, 2 Marks)

Q.4 What do you mean by priority queue? Explain the types to maintain the priority queue in memory. (AKTU 2018-19, 7 Marks)

Q.5 Define priority queue? Give one application of priority queue. (AKTU 2019-20, 2 Marks)

Q.6 Differentiate between circular queue and priority queue. (AKTU 2020-21, 2 Marks)

Q.7 Define priority queue and its uses. (AKTU 2021-22, 2 Marks)

GATEWAY CLASSES

Priority Queue

- Many applications requires priority queues rather than the simple FIFO strategy.
- A priority queue is another type of queue data structure in which elements can be inserted or deleted based on the priority i.e. each element in the priority queue is assigned a priority number and the element are inserted on the priority bases instead of using the order of arrival.
- If the priority is same than the FIFO order is used.

GATEWAY CLASSES

View of Priority Queue

FRONT PQ ↓ REAR PQ ↓

Elements			A	B	C	.....	X	.....	Z
Priority No.			P_1	P_2	P_3		P_i		P_n

GATEWAY CLASSES

Types of Priority Queue

There are two types of priority queues:-

1. Ascending Priority Queues
2. Descending Priority Queues

In ascending priority queue elements can be inserted in an order. But while deleting elements from the queue, always an element with small priority number to be deleted first.

In descending priority queue elements can also be inserted in any order but while deleting elements from the queue always a number with larger priority number will be deleted first.

GATEWAY CLASSES

Representation of the Priority Queue

There are various ways of maintaining a priority queue.

These are-

- One way linked list
- Multiple queues, one queue for each priority
- Maximum and Minimum Heap (Maxheap, Minheap) } Tree

Tree → Complete Binary Tree

GATEWAY CLASSES

One way list representation

In this representation each node of the linked list will have 3 fields.

1. An information field i.e. INFO
2. A priority number i.e. PRN
3. An address to the next node i.e. LINK

INFO	PRN	NEXT

NODE REPRESENTATION TO REPRESENT PRIORITY Queue through one-way list representation

- A node X precedes a node Y in the list when X has higher priority than Y or when both have same priority but X was inserted to the list before Y.
- Priority number will operate in the usual way i.e. the lower the priority number, the higher the priority.

GATEWAY CLASSES

The first element in the list is the element with maximum priority. So it is very simple to delete an element from the priority queue. It takes only O(1) time.

Algorithm: PRIORITY QUEUE DELETION

1. ITEM = INFO[FIRST]
2. DELETE FIRST NODE FROM THE LIST
3. PROCESS ITEM
4. STOP

GATEWAY CLASSES

Adding an element to priority queue is much more complicated then deleting an element from the queue, because we need to find the correct place to insert the element.

Suppose we add an ITEM with priority number N to a priority queue than-

(a) Traverse the list until finding a node X whose priority number exceeds N.

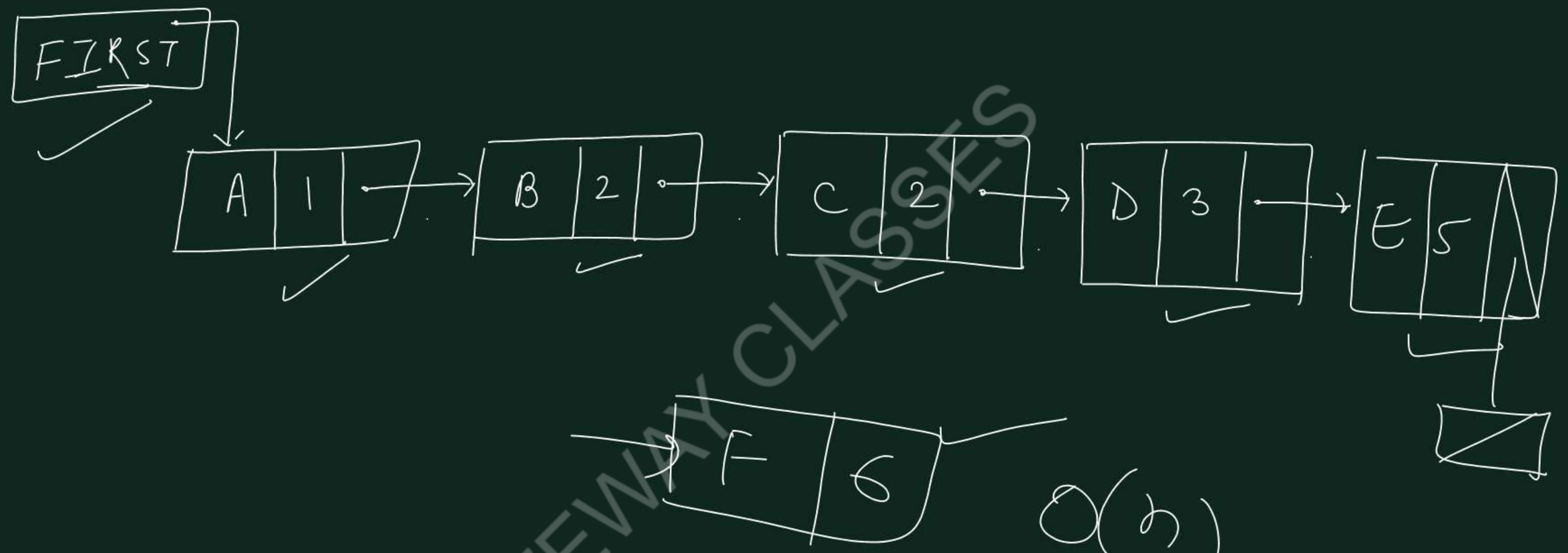
(b) Insert the node ITEM in front of X.

(c) If no such node is found , insert ITEM as the last node of the list.

So, insert needs O(n) time.

Consider the following priority queue with 5 nodes.

GATEWAY CLASSES

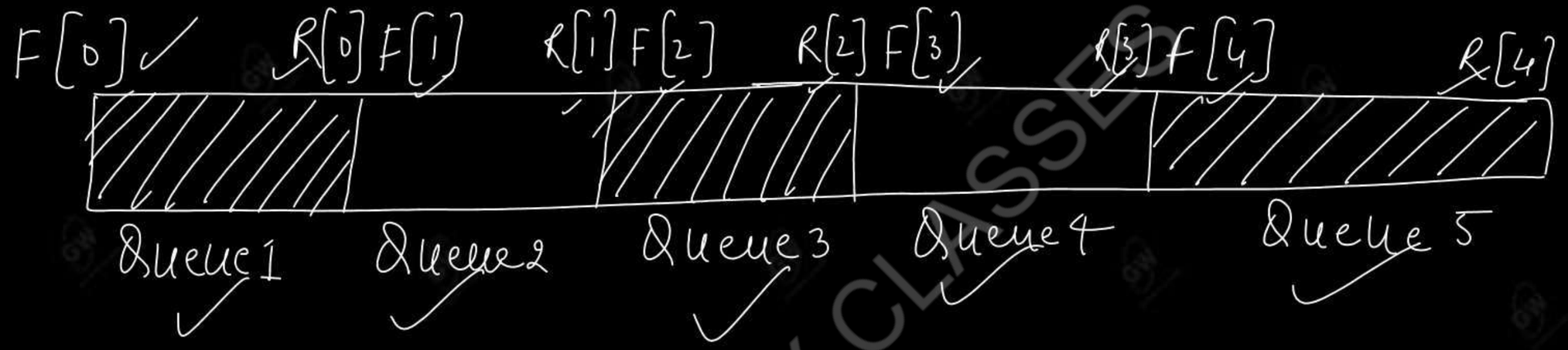


$O(n)$

GATEWAY CLASSES

Multiple Queue Representation of Priority Queue

Consider the following representation:-



- In this representation, multiple queues are used in a same array queue[] of size n.
- Each queue i is allocated a predefined space bounded by array indices.
- In order to preserve multiple queues, there should be multiple front and multiple rear ends. So, here a vector front and rear denotes the indices for each queue.

GATEWAY CLASSES

Heap Representation of a Priority Queue

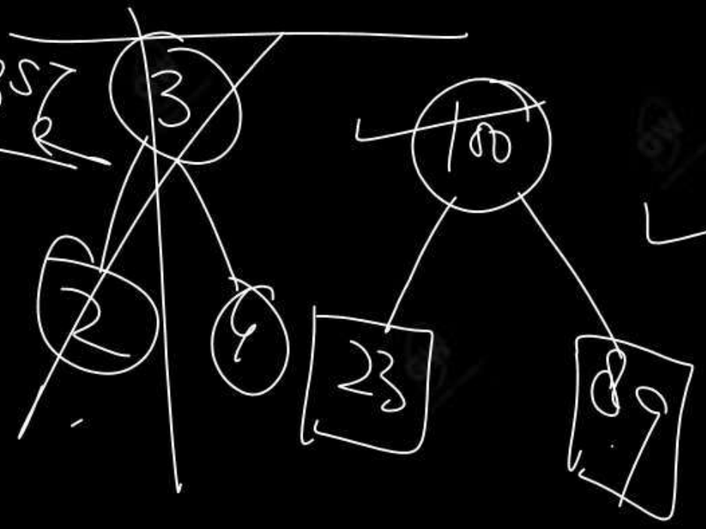
- A heap is a complete binary tree with additional property – the root node is either smallest or largest from its children. → lar Maxheap.
- If root node of the heap is smallest from its children, it is called min heap. ↑ min heap
- If root node is largest from its children, it is called max heap.
- A priority queue having highest priority for lower number can be represented using min heap and a priority queue having the highest priority for higher number can be represented using max heap.

Heap

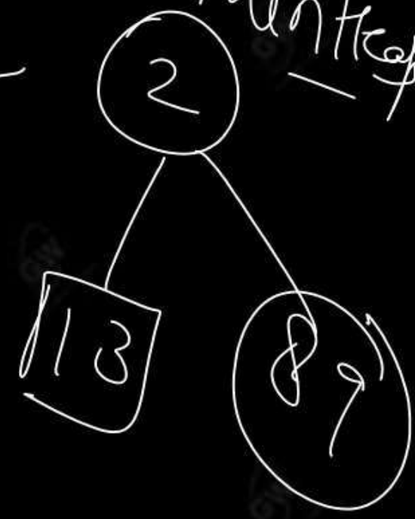
Max

Min

BST



Min Heap



GATEWAY CLASSES

Applications of Queue Data Structure

2 marks

- There are several algorithms that use queues data structure to solve problems easily. For example, BFS, traversing of a binary tree etc.
- Round – Robin scheduling for processor scheduling is implemented using queues.
- When the jobs are submitted to a networked printer, they are arranged in order of arrival i.e. jobs sent to a printer are placed on a queue.
- Every real life line is a queue. For example, lines at ticket counters at cinema hall, railway stations, bus stands etc., are queues because the service , i.e., ticket is provided on first come first served (FIFO) basis.

GATEWAY CLASSES

Difference between Stack and Queues

Stacks	Queues
1. Stack is an ordered list where in all insertions and deletions are performed at one end called top. PUSH, POP Top of the stack	1. Queue is an ordered list where in insertions are performed at rear end and deletions are performed at front end. front = Deletion, Rear = Insertion
2. Stack is called LIFO structure.	2. Queue is called FIFO.
3. Associated with stack there is one variable called top. TOS	3. Associated with queue there are two variable called front and rear.

GATEWAY CLASSES

Difference between Stack and Queues

Stacks	Queues
4. Stack is full can be represented by the condition Top = MAXSIZE-1	4. Queue is full can be represented by the condition rear = MAXSIZE-1
5. Stack is empty is represented by the condition Top = -1	5. Queue is empty is represented by the condition front = -1.
6. To insert an element into the stack top is incremental by 1.	6. To insert an element into the queue rear is incremented by 1.
7. To delete an element from the stack Top = Top – 1.	7. To delete an element from the queue front is incremented by 1.

GATEWAY CLASSES

Download Gateway Classes Application
From Google Play store
Link in Description

Next Lecture:- Iteration and Recursion

Thank You

GATEWAY CLASSES

UNI-2

1. Stack
2. Queue
3. Iteration & Recursion

GATEWAY CLASSES

B.Tech

Data Structure

Unit- 2: Lec-8

CS / IT / CS Allied

By- Dr. Kapil Kumar
M.Tech.(CSE), Ph.D.(CSE)

GATEWAY CLASSES

Today's Target

- What is Recursion?
- What are the properties of Recursive Function?
- Calculation of Factorial of a number through Iteration
- Calculation of Factorial of a number through Recursion
- Why use Recursion and how this approach is different from the Iteration
- What are the Principles of Recursion
- Write following programs in C using Recursion:-

(a) To print the reverse of a multi-digit number

(b) To find the sum of all digits of a multi-digit number

(c) To multiply two natural numbers

(d) To print the Fibonacci series

(e) To calculate the power of a given number

AKTU PYQs

GATEWAY CLASSES

AKTU PYQs

Q.1 What is recursion? Write a recursive program to find the sum of digits of a given number. Also calculate the time complexity. [ex:259=16=7(answer)].

AKTU 2016-17, 10 Marks

Q.2 What is recursion? A recursive procedure should have two properties. What are they?

AKTU 2016-17, 2 Marks

Q.3 Define the recursion. Write a recursive and non recursive program to calculate the factorial of a given number.

AKTU 2017-18, 7 Marks

Q.4 Write a C program to multiply 2 integer numbers using recursion.

AKTU 2017-18, 2 Marks

Q.5 What is recursion? Give disadvantages of recursion. AKTU 2018-19, 2 Marks

Q.6 (i) What is recursion?

(ii) Write a C program to calculate the factorial of number using recursive and non recursive functions.

AKTU 2021-22 10 Marks

Q.7 Which data structure is used for recursion and why? AKTU 2022-23, 2 Marks

GATEWAY CLASSES

Recursion

- When a function call to itself that function is known as recursive function.
- Recursion is a process of expressing a function in terms of itself.
- We can also define recursion as a process in which a function calls itself with reduced input and has a base condition to stop the process, i.e. in order to solve a problem recursively, the following two conditions must be satisfied:-

(a) First, the problem must be written in a recursive form, and

(b) Second the problem statement must include a stopping condition.

GATEWAY CLASSES

```
main()
{
}
```

1) Function protype or function declaration

2) function call

3) function Definition

GATEWAY CLASSES

/* Funⁿ Defⁿ */

```
int sum(int a, int b)
{
    int c;
    c = a+b;
    return c;
}
```

5 6

function header

5 6

~~2 a, 2 b~~

```
c = sum(a,b);
```

GATEWAY CLASSES

/* funⁿ Defⁿ */

void ABC( )

{

ABC( ); /* funⁿ call statement */

}

GATEWAY CLASSES

Recursion

Stack framming

- When a function is called, the return address, the values of local and formal variables are pushed onto the stack, a block of memory of contiguous locations, set aside for this purpose.
- After this the control enters into the function.
- Once the return statement is encountered, control comes back to the previous call, by using the return value present in the stack, and it substitutes the return value to the call.
- If the function does not return any value, control goes to the statement that follows the function call.

GATEWAY CLASSES

Example:1 Write a program in C to calculate the factorial of a number using iteration.

```
{
int a;
int fact=1;
printf("\n Enter a no:");
scanf("%d", &a);
while(a != 0)
{
fact = fact * a;
a--;
}
printf fact;
}
```

→ 120

fact = 5 * 4
= 20
a = 3

3 != 0 T
fact = 20 * 3
= 60
a = 2

2 != 0 T
fact = 60 * 2
= 120
a = 1

1 != 0 T
fact = 120 * 1 = 120
a = 0

0 != 0 F

Output:-
5! = 5 * 4 * 3 * 2 * 1
= 120

Input : 5

fact = 1
a = 5
5 != 0 T
fact = 1 * 5 = 5
a = 4

4 != 0 T

GATEWAY CLASSES

```
int factorial ( int );
void main()
{
    int a;
    int fact;
    print ———
        &a;
    fact = factorial (a);
    printf(" In factorial of %d=%d", a, fact);
    getch();
}
```

```
int factorial (int a)
{
    int fact=1;
    while ( a != 0)
    {
        fact = fact * a;
        a--;
    }
    return fact;
}
```

GATEWAY CLASSES

Example:2. Write a program in C to calculate the factorial of an integer using recursion.

Solution:-

```
/*TO CALCULATE THE FACTORIAL OF A NUMBER USING RECURSION*/

int factorial(int);   /* fun' prototype */
#include<stdio.h>
void main()
{
int a,fact;
clrscr();
printf("\n enter a number to calculate the factorial of that number:");
scanf("%d",&a);
fact=factorial(a);   /* funcall */
printf("\n factorial of %d=%d",a,fact);
getch();
}

int factorial(int a)
{
if(a==0)
return 1;
else
return(a*factorial(a-1));
}
```

Base criteria

5 == 0 F
4 == 0 F
3 == 0 F
2 == 0 F
1 == 0 F
0 == 0 T

5 = 120

a = 5
5 * factorial(4)
5 * 4 * factorial(3)
5 * 4 * 3 * factorial(2)
5 * 4 * 3 * 2 * factorial(1)
5 * 4 * 3 * 2 * 1 * factorial(0)
5 * 4 * 3 * 2 * 1 * 1
= 120

GATEWAY CLASSES

Definition & Principle of Recursion:-

- A function is said to be recursively defined, if a function containing either a Call statement to itself or a Call statement to a second function that may eventually result in a Call statement back to the original function.

Direct

A recursive function must have the following properties:-

1. There must be certain criteria, called base criteria for which the function does not call itself.
2. Each time the function does call itself (directly or indirectly), the argument of the function must be closer to a base value.

GATEWAY CLASSES

① Direct

ABC()

{

ABC()

}

② Indirect Recursion

ABC()

{

XYZ()

{

XYZ()

{

ABC()

}

GATEWAY CLASSES

WHY USE RECURSION AND HOW THIS APPROACH IS DIFFERENT FROM ITERATION

- Programming in interactive method helps the programmer to perform certain tasks repeatedly for a known number of times or as long as the desired condition is true using the for loop or while loop or do-while loop. But our interest is to make a function execute repeatedly in terms of itself. So this mechanism is called Recursion.
- Recursion is a very powerful technique to write a complicated program in a easy way. This is an important facility available in most of the structured programming languages such as Pascal, C, C++ etc.

GATEWAY CLASSES

Example 3:- Write a program in C to print the reverse of a multi-digit number using recursion.

Solution:-

Note:- The static variables are able to preserve their previous value, and one doesn't need to initialize them again.

```
int reverse_digit(int);
#include<stdio.h>
void main()
{
int num,result;
clrscr();
printf("\n enter a multi-digit number:");
scanf("%d",&num);
result=reverse_digit(num);
printf("\n The reverse of the number
is:%d",result);
getch();
}
int reverse_digit(int a)
{
int b;
static int r=0;
if(a==0)
return a;
else
{
b=a%10;
r=r*10+b;
reverse_digit(a/10);
}
return(r);
}
```

/* funⁿ prototype */

1243

3421

funⁿ Defn

Input:- 1243

Output:- 3421

1234==0 F
123==0 F
12==0 F
1==0 F
0==0 T

1243/10=124
124/10=12
12/10=1
1/10=0

GATEWAY CLASSES

$a = 1243$

$r = 0$

$b = a \% 10 = 1243 \% 10 = 3$

$r = r * 10 + b$

$= 0 * 10 + 3 = 3$

$a = 124$

$b = a \% 10 = 124 \% 10 = 4$

$r = r * 10 + b = 3 * 10 + 4$

$= 34$

$a = 12$

$b = a \% 10 = 12 \% 10 = 2$

$r = r * 10 + b = 34 * 10 + 2$

$= 340 + 2 = 342$

$10 \overline{)1243(124}$

10

24

20

43

40

3

quotient

float ✗

remainder

$a = 1$

$b = 1 \% 10 = 1$

$r = 342 * 10 + 1$

$= 3420 + 1$

$= 3421$

GATEWAY CLASSES

Storage classes

- auto — By default
- static
- extern
- register

GATEWAY CLASSES

Example 4:- Write a program in C to find the sum of all digits of a multi-digit number using recursion.

Solution:-

```
int sum_digits(int);
#include<stdio.h>
void main()
{
int num,result;
clrscr();
printf("\n enter a multidigit number:");
scanf("%d",&num);
result=sum_digits(num);
printf("\n The reverse of the number
is:%d",result);
getch();
```

sum

/* fun^n call statement */

10

Input:- 1243

Output:- 3421

output:- 1+2+4+3 = 10

```
}
int sum_digits(int a)
{
int b;
static int sum=0;
if(a==0)
return a;
else
{
b=a%10;
sum=sum+b;
sum_digits(a/10);
}
return(sum);
}
```

1243

function header

1243==0 F
124==0 F
12==0 F
1==0 F
0==0 T

a=1243
sum=0
b=1243%10=3
sum=0+3=3
sum_digits(1243/10)
124

a=124
b=124%10=4
sum=3+4=7
sum_digits(124/10)
12

a=12
b=12%10=2
sum=7+2=9
sum_digits(12/10)
1

GATEWAY CLASSES

$a = 1$

$b = a \% 10 = 1 \% 10 = 1$

$sum = 9 + 1 = 10$

sum_digits (1/10);

0

$a = 0$

GATEWAY CLASSES

2 marks

Example -5 Write a program in c to multiply two natural numbers using recursion.

Solution:-

```
int multiply(int,int); /* func^n Prototype */
#include<stdio.h>
void main()
{
int num1,num2,result;
clrscr();
printf("\n enter the first number:");
scanf("%d",&num1);
printf("\n enter the second number:");
scanf("%d",&num2);
result=multiply(num1,num2);
printf("\n The multiplication of %d and %d =
%d",num1,num2,result);
getch();
}
int multiply(int i, int j)
{
if(i==0, j==0)
return 0;
if(j==1)
return i;
else
return(multiply(i,j-1)+i);
}
```

5 X 4 = 20

5 4 20 5 4 20

5 1
5 3
5 4

return(multiply(5,3)+5)
return(multiply(5,2)+5+5)
return(multiply(5,1)+5+5+5)
5+5+5+5
20

GATEWAY CLASSES

Example 6: Write a program in C to print the Fibonacci series using recursion.

Solution:-

```
void fibonacci_series(int);  /* funⁿ Prototype or funⁿ Declaration */
#include<stdio.h>
void main()
{
int counter,first=0,second=1;
clrscr();
printf("\n how many first term of fibonacci series you want to print:");
scanf("%d",&counter);  5
if(counter>2)  T
{
printf("\n %d",first);
printf("\n %d",second);  3
fibonacci_series(counter-2);
}
else if(counter==2)
{
printf("\n %d",first);
printf("\n %d",second);
}
else if(counter==1)
printf("\n %d",first);
else
printf("\n Fibonacci series printing is not possible");
getch();
}
```

0
1

GATEWAY CLASSES

```
void fibonacci_series(int a)
{
int b;
static int first=1,second=1;
if(a==0)
printf("\n The series ends here");
else
{
b=first+second;
first=second;
second=b;
printf("\n %d",b);
fibonacci_series(a-1);
}
}
```

Fibonacci series -

0, 1, 1, 2, 3, 5, ...

I 5 terms

Output :

0, 1, 1, 2, 3

0
1
1
2
3

The series ends here

a = 3 first = 0 second = 1

b = 0 + 1 = 1

first = 1

second = 1

a = 2

b = 1 + 1 = 2

first = 1

second = 2

b = 1 + 2 = 3

first = 2

second = 3

GATEWAY CLASSES

Example – 7: Write a program in C to calculate the power of a given number using recursion.

```c
int power(int,int);
#include<stdio.h>
void main()
{
int a,b,result;
clrscr();
printf("\n enter a base value:");
scanf("%d",&a);
printf("\n enter a power value:");
scanf("%d",&b);
result=power(a,b);
printf("\n The %d raised to %d =%d",result);
getch();
}
int power(int a,int b)
{
static int r=1;
if(b==0)
return 1;
else
{
r=r*a;
power(a,b-1);
}
return(r);
}
```

/* funⁿ prototype */

$3^3 = 27$

GATEWAY CLASSES

AKTU PYQs

Q.1 What is recursion? Write a recursive program to find the sum of digits of a given number. Also calculate the time complexity. [ex:259=16=7(answer)].

AKTU 2016-17, 10 Marks

Q.2 What is recursion? A recursive procedure should have two properties. What are they?

AKTU 2016-17, 2 Marks

Q.3 Define the recursion. Write a recursive and non recursive program to calculate the factorial of a given number.

AKTU 2017-18, 7 Marks

Q.4 Write a C program to multiply 2 integer numbers using recursion.

AKTU 2017-18, 2 Marks

Q.5 What is recursion? Give disadvantages of recursion. AKTU 2018-19, 2 Marks

Q.6 (i) What is recursion?

(ii) Write a C program to calculate the factorial of number using recursive and non recursive functions.

AKTU 2021-22 10 Marks

Q.7 Which data structure is used for recursion and why? AKTU 2022-23, 2 Marks

GATEWAY CLASSES

Download Gateway Classes Application
From Google Play store
Link in Description

Next Lecture:- ITERATION AND RECURSION 2:- Advantages and disadvantages of recursion, Difference between iteration and recursion, Recursion through stack etc.

Thank You

GATEWAY CLASSES

B.Tech

Data Structure

Unit- 2: Lec-9

CS / IT / CS Allied

By- Dr. Kapil Kumar
M.Tech.(CSE), Ph.D.(CSE)

GATEWAY CLASSES

Today's Target

- Write a program in C to print the table of a number using recursion.
- Write a program in C to reverse an array of n elements using recursion.
- Advantages and Disadvantages of Recursion
- Difference between Iteration and Recursion
- Recursion Through Stack
- Recursive Function : Greatest Common Divisor (GCD) HCF
- Write a C program to find the GCD of two numbers using recursion.
- Direct and Indirect Recursion
- AKTU PYQs

GATEWAY CLASSES

AKTU PYQs

Q.1 What is recursion? Give disadvantages of recursion.

AKTU 2018-19, 2 Marks

Q.2 Differentiate between iteration and recursion.

AKTU2019-20, 5 Marks

Q.3 Discuss disadvantages of recursion with some suitable example.

AKTU2021-22, 5 Marks

Q.4 Which data structure is used for recursion and why?

AKTU 2022-23, 2 Marks

GATEWAY CLASSES

Example 1:- Write a program in C to print the table of a number using recursion.

Solution –

```
int table ( int , int );
#include<stdio.h>
void main()
{
int a,b,result;
clrscr();
printf("\n enter a number for which the table is to be printed:");
scanf("%d",&a);
for(i=1;i<=10;i++)
printf("\n %d * %d = %d",a,i,table(a,i));
getch();
}
int table(int a,int i)
{
if(i==1)
return a;
else
return(table(a,i-1)+a);
}
```

5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
10

5 X 1 =
5 X 2 3

table (5,1) + 5
5 + 5 = 10

table (5,2) + 5
table (5,1) + 5 + 5
5 + 5 + 5 = 15

GATEWAY CLASSES

Example 2 : - Write a program in C to reverse an array of n elements using recursion.

Solution:-

Input:-

0	1	2	3	4
11	22	33	44	55

Output:

0	1	2	3	4
55	44	33	22	11

```
void main()
{
int a[20],i,n;
printf("\n How many numbers you want to input:");
scanf("%d",&n);
if(n>20)
{
printf("\n Invaid input, size must be less then 20");
getch();
exit();
}
/*Input array for n elements */
for(i=0;i<n;i++)
{
printf("\n enter number %d:",i+1);
scanf("%d",&a[i]);
}
printf("\n the reverse array is:");
reverse(a,n-1);  /* function call */
getch();
}
```

5-1
4

GATEWAY CLASSES

```
void reverse(int a[],int n)
{
If(n==-1)
return;
printf("\n%d",a[n]);
reverse(a,n-1);
}
}
```

4 --- -1 F

-1 --- -1 T

else {

/* fun call */

addr

Output -

a [4] = 55

[3] = 44

[2] = 33

[1] = 22

[0] = 11

GATEWAY CLASSES

ADVANTAGES AND DISADVANTAGES OF RECURSION

Advantages:

Some advantages of using recursion are:

1. We can create a simple and easy version of programs using recursion.
2. Always recursion will be written in the name of recursive definition. It can be translated into C code very easily.
3. We can avoid initialization of variable inside the functions, but iterative solutions are required to be initialized.
4. Some specific applications are meant for recursion such as Binary tree traversal, tower of Hanoi etc. can be easily understood.

GATEWAY CLASSES

TREE

TRAVERSAL

- Preorder traversal
- Inorder
- Postorder
- Level by level

Tail recursion

Tower of Hanoi

GATEWAY CLASSES

ADVANTAGES AND DISADVANTAGES OF RECURSION

Some disadvantage of recursion are as follows:-

1. **It occupies lot of memory:-** When function is called outside or called within, the function stores formal parameters, local variables and returns address to confirm function are working well.

2. **It consumes more time to get desired result:** After matching the base condition, the function should restore the most recently saved parameters, local variables, and return address. This operation spends lot of time during pushing and popping the necessary items to and from the stack.

3. Function execution is slower than iterative method because of the overhead of calling functions repeatedly.

Therefore, recursion is expensive in terms of processor time and memory usage.

GATEWAY CLASSES

Difference between Iteration and Recursion

Any iterative function consists of four parts:-

1. Initialization: The decision parameter is set to an initial value or more precisely, the loop variable is initialized, other pre-defined variables are also initialized.

2. Decision: The decision parameter is used to determine whether further looping is necessary. The loop variable is compared and based on the outcome, the loop is executed again.

3. Computation: Necessary computation is performed within the loop.

4. Update: The decision parameter is updated and a transfer is made to the next iteration. The loop variable is incremented/decremented.

GATEWAY CLASSES

```
for ( initialize ; condition ; inc/dec )
{
    Body ;
}
```

1 2 3 4

GATEWAY CLASSES

For a Recursive Function:-

1. **Initialization:** The variables in the form of arguments are passed on to the function.
2. **Decision:** The argument values are used to determine whether further recursive calls are required.
3. **Computation:** Necessary computation is performed using the local variables and the parameters at the current depth.
4. **Update:** The update is done so that the variables can be used for further recursive calls.

GATEWAY CLASSES

Iteration

1. It is a process of executing or group of statements repeatedly until some specified condition is satisfied. Here counter control the execution and body of the loop terminates when the termination condition is failed (false).

For Example:-

```
int fact (int n)
{
int i, prod = 1 ;
for ( i = 1 ; i < n ; i ++)
 prod = prod * I ;
return(prod) ;
}
```

Recursion

1. It is the name gives to the technique of defining something in terms of itself. It is terminated, when a base condition is met. The terminal condition is gradually reached, by invocation of the same function. Each time function is called itself till the base condition is reached.

For Example:-

```
int fact (int n)
{
if (n = = 0)
        return 1 ;
else
    return (n*fact (n-1));
}
```

GATEWAY CLASSES

Iteration	Recursion
2. This process involves some clear cut processing sections such as initialization, decision, computation and updation (of iteration counter).	2. There will be an exclusive stopping condition within the recursive body of the function.
3. It uses repetitions structures such as for loop, while-loop and do-while loop and explicitly uses repetition structures.	3. It uses selection structures such as if-statement, if-else or switch statement and achieves repetition through function calls itself repeatedly.

GATEWAY CLASSES

4. These functions execute very fast and occupy less memory and it can be easy.	4. In recursive, every time the function is called, all the local variables, formal parameters and return address are pushed onto the stack. So it occupies lot of memory (stack) and it occupies more memory because more number of times pushing and poping items. Therefore, recursion, is expensive in terms of processor time and memory usage.
5. There are some applications in which iteration is not best suited such as tower of Hanoi, tree traversals etc. These functions are very difficult to design, takes more programmers time and more error prone.	5. There are some applications in which recursion is best suited for designing algorithms such as tower of Hanoi, Binary Tree traversals etc. In this case, these functions are more efficient and can be understood easily.

GATEWAY CLASSES

RECURSION THROUGH STACK

- Some programming languages like 'C' provide the facility of recursive function. So we can use it very easily. But in other language also we can provide recursion technique with stack implementation whenever the situation arises for recursion implementation.
- Even we can convert recursive function in 'C' into non-recursive function through stack implementation.
- It can be seen in graph traversal where we use Depth first traversal using recursion as well as stack implementation.

GATEWAY CLASSES

RECURSION THROUGH STACK

- We have a need to do following things for using recursion through stack implementation.

1. One stack for each parameter of function.
2. Only one stack for return address.
3. One stack for each local variable of function.

- We have a need to take right decision where to use recursion function or stack implementation because most of the places using stack implementation will be complex and takes more execution time but in some situations it can be more useful than recursion also.

GATEWAY CLASSES

Greatest Common Divisor (GCD) of Two numbers

- The greatest common divisor of two integers m and n is the largest common divisor of m and n and is denoted by GCD (m, n). This divides both m and n such that the remainder is zero.
- The greatest common divisor (GCD), also known as "Greatest Common Factor (GCF)" or "Highest Common Factor (HCF)", of two numbers is the greatest factor that divides both the numbers.
- Euclid's Method: To find the greatest common divisor of two positive integers with the help of Euclid's algorithm is as follows:
- The recursive definition for finding an GCD of two numbers is:-

$$GCD(m,n) = \begin{cases} GCD(n,m) & if\ n > m \\ m & if\ n = 0 \\ GCD(n, m \bmod n) & if\ m > n \end{cases}$$

GATEWAY CLASSES

Example 3:- Write a C program to find the GCD of two numbers using recursion.

Solution:-

```
int gcd(int,int);
#include<stdio.h>
void main()
{
int a,b,result;
clrscr();
printf("\n enter the first number:");
scanf("%d",&a);
printf("\n enter the second number:");
scanf("%d",&b);
result=gcd(a,b);
printf("\n The gcd of %d and %d =
%d",a,b,result);
getch();
}
int gcd(int a,int b)
{
int result;
if(b==0)
return a;
else
{
result=a%b;
return(gcd(b,result));
}
}
```

GATEWAY CLASSES

EXAMPLE 4. Write a recursive algorithm to implement the following function:-

Ackerman function

$$A(m,n) = \begin{cases} n+1 & if\ m = 0 \\ A(m-1,1) & if\ m\ != 0\ and\ n = 0 \\ A(m-1, A(m, n-1)) & if\ m\ != 0\ and\ n\ != 0 \end{cases}$$

And considering this algorithm, find A (1 , 3). {Ans = 5}

Solution:-

Algorithm A(m,n)

1. If m==0

Return n + 1

2. If n==0

Return call A(m-1,1)

3. Call A((m-1,A(m,n-1))

4. Stop

GATEWAY CLASSES

$$A(m,n) = \begin{cases} n+1 & if\ m = 0 \\ A(m-1,1) & if\ m\ != 0\ and\ n = 0 \\ A(m-1, A(m, n-1)) & if\ m\ != 0\ and\ n\ != 0 \end{cases}$$

Considering this algorithm ,find A (1 , 3). {Ans = 5}

Solution:-

$A(\overset{m}{1}, \overset{n}{3}) = A(0, A(1,2))$

$A(1,2) = A(0, A(1,1))$

$A(1,1) = A(0, A(1,0))$

$A(1,0) = A(0,1)$

$A(0,1) = 2$

$A(1,0) = 2$

$A(1,1) = A(0,2)$

$A(0,2) = 3$

$A(1,1) = 3$

$A(1,2) = A(0,3)$

$A(0,3) = 4$

$A(1,2) = 4$

$A(1,3) = A(0,4)$

$A(0,4) = 5$

Ans $A(1,3) = 5$

GATEWAY CLASSES

EXAMPLE 5: Let a and b are positive integers. Suppose a function Q is defined recursively as follows:

$$Q(a,b) = \begin{cases} Q & \text{if } a < b \\ Q(a-b,b)+1 & \text{if } b \le a \end{cases}$$

Find the value of Q(2,3) and Q(14,3). (ans =0,4)

(i) $Q(\overset{a}{2},\overset{b}{3}) = 0$

(ii) $Q(14,3) = Q(11,3)+1$

$= Q(8,3)+1+1$

$= Q(5,3)+1+1+1$

$= \underbrace{Q(\overset{a}{2},\overset{b}{3})}_{0}+1+1+1+1$

$= 0+4$

$= 4$

GATEWAY CLASSES

EXAMPLE 6: A recursive function f is as follows:-

```
int f ( int x )
{
if ( x < 2 )
return 1;
else
return f ( x – 1 ) + f ( x – 2 );
}
```

Find value of f (5). (ans=8)

Solution:-

F

f(5)

return f(4) + f(3)

return f(3) + f(2) + f(2) + f(1)

f(2) + f(1) + f(1) + f(0) + f(1) + f(0) + 1

f(1) + f(0) + 1 + 1 + 1 + 1 + 1 + 1

1 + 1 + 6

= 8

GATEWAY CLASSES

DIRECT AND INDIRECT RECURSION

- Recursion is said to be direct if a subprogram calls itself, it is indirect if there is a sequence of more than one sub-program call that eventually calls the first sub-program such as when function A calls function B, which in turn calls A.

(a) ABC()

(b) ABC()

GATEWAY CLASSES

AKTU PYQs

Q.1 What is recursion? Give disadvantages of recursion.

AKTU 2018-19, 2 Marks

Q.2 Differentiate between iteration and recursion.

AKTU2019-20, 5 Marks

Q.3 Discuss disadvantages of recursion with some suitable example.

AKTU2021-22, 5 Marks

Q.4 Which data structure is used for recursion and why?

AKTU 2022-23, 2 Marks

GATEWAY CLASSES

Download Gateway Classes Application
From Google Play store
Link in Description

Next Lecture:- Tail Recursion and Tower of Hanoi Problem

Thank You

GATEWAY CLASSES

B.Tech

Data Structure

Unit- 2: Lec-10

CS / IT / CS Allied

By- Dr. Kapil Kumar
M.Tech.(CSE), Ph.D.(CSE)

GATEWAY CLASSES

Today's Target

- Tail Recursion
- Tower of Hanoi Problem
- AKTU PYQs

GATEWAY CLASSES

TAIL RECURSION

- As we know the process in which function calls itself in function body is called as recursion. But when this called function is the last executed statement in the function body then it is called "tail recursion".
- We use stack to maintain all the functions but here it will not append new function in stack but it will overwrite the value of previous function with the current one.
- So we can say the function call time and stack implementation time will be reduced and it will give better performance.

GATEWAY CLASSES

TAIL RECURSION

- In tail recursion, there is a single recursive call that occurs at the very end of the function implementation.
- In other words, when the recursive call is made, there are no more statements to be executed in the function. The single recursive call is the last thing to happen when the function is executed.

GATEWAY CLASSES

Consider the following example:-

```
void main()
{
function(3);
getch();
}
void function(int n)
{
if(n==0)
return;
else
printf("%d",n);
return function(n-1);
}
```

321

3 2 1 0

F F

2 1 0

GATEWAY CLASSES

THE TOWERS OF HANOI PROBLEM OR

THE PROBLEM OF BANARAS TAMPLE OR

TOWER OF BRAHMA OR

LUCAS TOWER

- Tower of Hanoi is a mathematical puzzle invented by French Mathematician Edouard Lucas in 1883.
- The problem has a historical basis in the ritual of the ancient Tower of Brahma.
- History of Tower of Hanoi:-
- Numerous myths regarding the ancient and mystical nature of the puzzle popped up almost immediately, including a myth about Kashi Vashwanath Temple containing a large room with three poles in it, surrounding by 64 golden disks.
- This story of Indian temple was spread tongue-in-check by a friend of Lucas.

GATEWAY CLASSES

- If the priests were able to move disks at a rate of one per second, using the smaller number of moves, it would take them $2^{64}-1$ seconds or roughly 585 billions (585 arabs) years to finish, which is about 42 times the estimated age of the universe. The universe is approximately 13.8 billion years old.

$2^{64}-1$

GATEWAY CLASSES

In the tower of Hanoi puzzle we have 3 rods and n disks and the objective of the puzzle is to move entire stack of disks from first rod to another rod by obeying the following rules:-

1. Only one disk can be moved at a time.
2. Each move consists of taking upper disk from one of the rods and placing it on top of another rod or an empty rod.
3. No disk can be placed on the top of the disk that is smaller it.

GATEWAY CLASSES

Consider the following figures with 3 rods and 3 disks.

GATEWAY CLASSES

- The requirement is, we have to shift disks from rod A to rod C using rod B by following above mentioned 3 rules.
- To understand this puzzle we will consider 3 disks as:-

GATEWAY CLASSES

Procedure:-

1. Move top n-1 disks from rod a to rod B (A->B)
2. Move disks from A to C.
3. Move n-1 disks from B->C.

GATEWAY CLASSES

0 1 2 3

Algorithm:- TOH (n,A,B,C)

Where n = number of disks

A = from

B= using

C= to

If (n > 0) T

{

0 1 3 2

TOH(n-1,A,C,B)

Print(From %d to %d",A,C);

TOH(n-1,B,A,C);

}

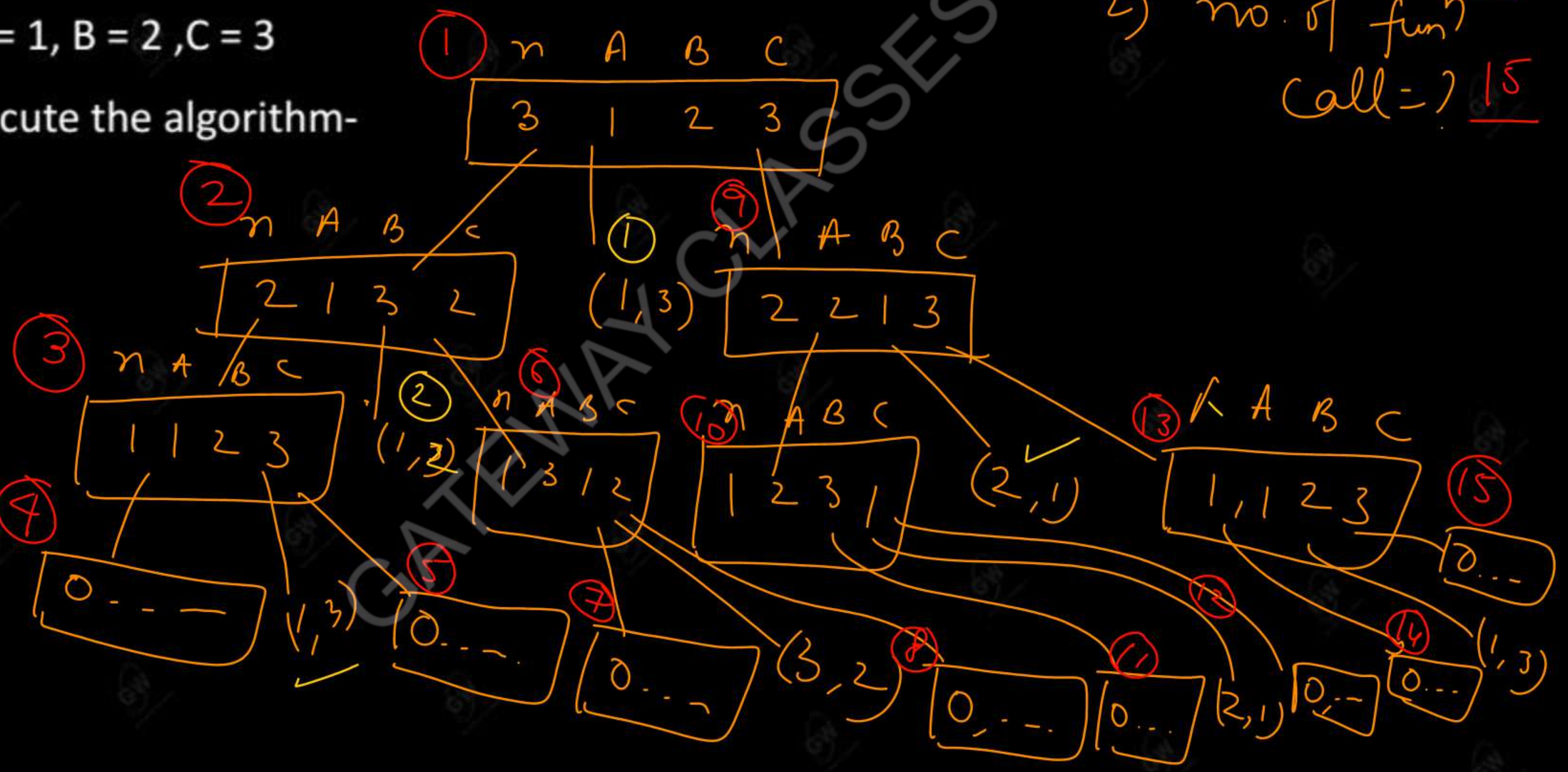
GATEWAY CLASSES

Here in our example –

N = 3, A = 1, B = 2 ,C = 3

Now execute the algorithm-

1) movement => 7
2) no. of func call => 15



GATEWAY CLASSES

In this example total number of function calls = 15

And sequence of movement is as follows:-

(1,3)

(1,2)

(3,2)

(1,3)

(2,1)

(2,3)

(1,3)

GATEWAY CLASSES

General Formula for number of function calls = $2^{n+1} - 1 = 2^{3+1} - 1 = 2^4 - 1$
$= 16 - 1$
$= 15$

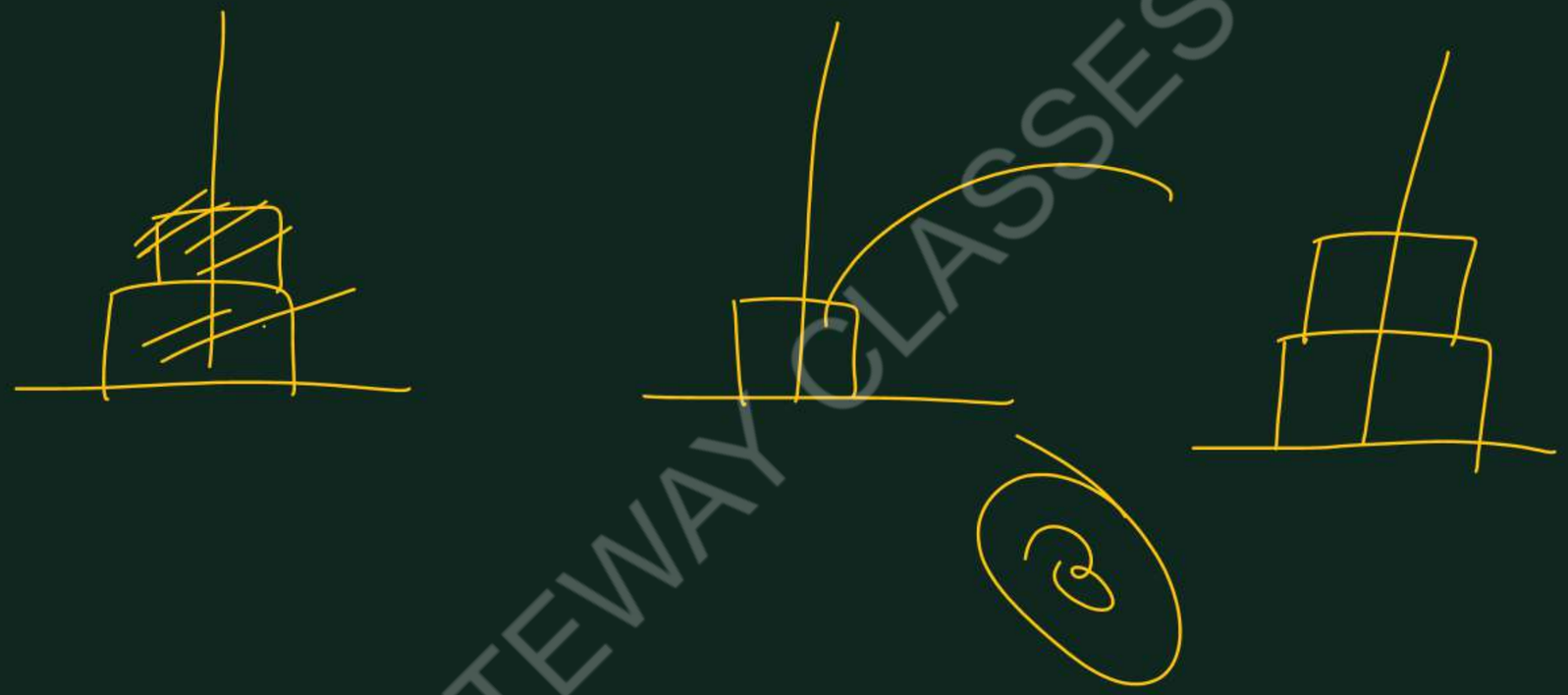
Formula for number of movements = $2^n - 1 = 2^3 - 1 = 8 - 1 = 7$

Example:- Say number of disks = 3

Example : Say number of disks = 2 $\longrightarrow$ no. of fun^n call $= 2^{n+1} - 1 = 2^{2+1} - 1$
$= 8 - 1 = 7$

no. of movement $= 2^n - 1$
$= 2^2 - 1$
$= 4 - 1 = 3$

GATEWAY CLASSES



GATEWAY CLASSES

AKTU PYQs

Q.1 If the Tower of Hanoi is operated on n=10 disks, calculate the total number of moves.

(AKTU 2015-16, 2 MARKS)

Q.2 Write an algorithm for finding solution to the tower of Hanoi problem. Explain the working of your algorithm (with 4 disks) with diagram.

(AKTU 2015-16, 15 MARKS)

Q.3 Write an algorithm for finding solution to the Towers of Hanoi Problem. Explain the working.

(AKTU 2016-15, 10 Marks)

Q.4 Calculate total number of moves for Tower of Hanoi for n=10 disks.

(AKTU 2017-18, 2 Marks)

Q.5 Write a recursive solution to Tower of Hanoi problem.

(AKTU 2018-19, 2 Marks)

GATEWAY CLASSES

Q.6 Explain Tower of Hanoi problem and write a recursive algorithm to solve it.

AKTU 2018-19, 7 Marks

Q.7 Explain Tail Recursion. AKTU 2019-20, 2 Marks

Q.8 Write the recursive solution for Tower of Hanoi problem. AKTU 2019-20 5 Marks

Q.9 Explain objective of Tower of Hanoi problem. AKTU 2020-21, 2 Marks

Q.10 Write a program for Tower of Hanoi problem with 'n' number of disks. Show all steps all steps used in solving Tower of Hanoi for 3 disks. AKTU 2020-21, 7 Marks

Q.11 What is tail recursion? Explain with a suitable example. AKTU 2021-22, 2 Marks

Q.12 What do you understand by tail recursion? AKTU 2022-23, 2 Marks

GATEWAY CLASSES

Download Gateway Classes Application
From Google Play store
Link in Description

GATEWAY CLASSES

Next Lecture:- Unit – 3, Searching (Linear and Binary)

Thank You