Page No.
Date : / /

4. Stack & Queue

Stack (FIFO (LIFO)

A stack is a list of element in which an element may be inserted or deleted only at one end called top of the stack.

The two operation related with stack are :

i) Push (insertion) :- use to insert an element into a stack

ii) pop (deletion) :- use to delete an element from the stack.

- Array representation of stack

This Procedure Pushes an item on to a stack

PUSH (Stack, top, MAXSTACK, ITEM)

max[size] of stack

① If, Top := MAXStack then
Print :- Overflow and return

② Set Top := Top + 1

③ Set Stack [Top] = item

④ return

Page No.
Date : / /

∴ POP(stack, TOP, ITEM)

① If ; Top := Null(0) then
Print : Underflow and
Return

② Set Item := stack [Top]

③ Set Top := Top - 1

④ Return

~~PUSH~~ ~~POP~~

Page No.
Date : / /

Q. Consider the following stack of character where stack is allocated N = 8 memory cell
STACK := A, C, D, E, K, —, —, —
Describe the stack for the following operation takes place.

1) POP (Stack, item)
⇒ A, C, D, E, —, —, —, —
2) POP (Stack, item)
A, C, D, —, —, —, —, —
3) PUSH (Stack, item)
A, C, D, L, —, —, —, —

4) Push (Stack, P)
A, C, D, L, P, —, —, —
5) POP (Stack item)
A, C, D, L, —, —, —, —
6) PUSH (Stack, R)
A, C, D, L, R, —, —, —
7) PUSH (Stack, Y)
A, C, D, L, R, Y, —, —

Q. Consider the following stack where stack is allocated N = 6 memory cell
Stack := AAA, DDD, FFF, FFF, GGG, —
Describe the stack as the following operation take place.

i) Push (Stack, KKK)

Page No.
Date : / /

i) POP (stack, item)
iii) Push (stack, LLL)
iv) Push (stack, NNN)
v) POP (stack, item)
vi) Push (stack, TTT)

Ans:- i) AAA, DDD, EEE, FFF, GGG, kkk
ii) AAA, DDD, EEE, FFF, GGG, ---
iii) AAA, DDD, EEE, FFF, GGG, LLL
iv) Overflow.
v) AAA, DDD, EEE, FFF, GGG, ---
vi) AAA, DDD, EEE, FFF, GGG, TTT

∴ Arithmetic expression.

let, Q be an arithmetic expression involving constant and operation. here we find the value of Q by using reverse polish (post fix notation) for that we assume following three level of precedence for the usual binary operation.

i) highest = Exponential (↑)
ii) next highest = Multiplication (*)
Division (/)
iii) lowest = Addition (+) &
Substraction (-)

Page No.
Date :

- Q ⇒ 2 ↑ 3 + 5 * 2 ↑ 2 - 12/6

Q ⇒ 8 + 5 * 2 ↑ 2 - 12/6

⇒ 8 + 10 5 * 4 - 12/6

⇒ 8 + 20 - 12/6

⇒ 8 + 20 - 2

⇒ 26

- Polish Notation

2 1 3

$= 8 + 5 \times 4 - 12/6$

$= 8 + 20 - 2$

$= 26$

- Polish Notation

1. $(A+B) \times C$

a. Prefix

$(+AB) * C$

$= * ABC$

b) Postfix

$(AB+) * C$

$= AB+C*$

2. $A + (B * C)$

a) Prefix

$\rightarrow A + (* BC)$

$+ A * BC$

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

a. Prefix

$\Rightarrow (+AB) / (-CD)$

$\rightarrow / (+AB) / (-CD)$

b) ⇒ A + (BC ★)
⇒ ABC ★ +

b) (AB+) / (CD-)
⇒ / (AB+) (CD-)

4) (A-B) ★ (D/E)

a) Prefix
(-AB) ★ (/DE)
★ (-AB) (/DE)

b) Postfix
(AB-) ★ (DE/)
(AB-)(DE/)★

5. (A+B ↑ D) / (E-F) + G

Postfix
⇒ (A+BD↑)(EF-)+G
= (ABD ↑+) / (EF-)+G
= (ABD↑+) / (EF-) + G
= (ABD↑+EF-/) + G
= ABD↑+EF-/G+

6 A★ (B+D) / E - F ★ (G+H/K)

b) Postfix
= A★ (BD+) / E-F ★ (G+HK/)
= A★ BD+ / E-F ★ GHK/+
= A★BD+E/-F ★ GHK/+
= ABD+E/★-FGHK/+★

= A★ (BD+) /E - F ★ (G+HK/)
= ABD+★/E-F ★ (GHK/+)
= ABD+★E/-FGHK/+★
= ABD+★E/FGHK/★-

Evaluation of Postfix Expression

This algorithm find the value of an arithmetic expression P written in postfix notation.

S1. Add a ")" at end of P.
S2. Scan P from left to right & repeat step + 3 & 4 for each each element of P until sentinel ")" is encountered.
S3. If an operand is encountered Put it on stack.
S4. If an operand ⊗ encountered then a) Remove top 2 element of stack, where A is top element & B is next top element
b) Evaluate b ⊗ A
c) Place the result of (b) on to the stack.
d) set value equal to top element on stack.
e) Exit.

1. Consider the following Arithmetic expression P given by

P → 5, 6, 2, +, *, 12, 4, /, -

P → 5, 6, 2, +, *, 12, 4, /, -

symbol scan	stack
5	5
6	5, 6
2	5, 6, 2
+	5, 8
*	40
12	40, 12
4	40, 12, 4
/	40, 3
-	37
)	

2. P ⇒ 12, 7, 3, -, /, 2, 1, 5, +, *, +,)

symbol scan	stack
12	12
7	12, 7
3	12, 7, 3
-	12, 4
/	3
2	3, 2
1	3, 2, 1
5	3, 2, 1, 5
+	3, 2, 6
*	3, 12
+	15
)	

Transformic Prefix Expression into Postfix:

Suppose Q is an Arithmetic Express written in Infix Notation this notation. Find the equivalent Postfix express P.

S1. Push "(" on to stack & add ")" to end of Q.

S2. Scan Q from left to right and repeat step 3 to 6 for each element of Q until stack is empty

S3. If an operand is encountered add it to P.

S4. If a left parenthesis is encountered push it on to stack.

S5. If an Q is encountered then
a) Repeatedly pop from stack & add to P each ⊗ which has the same presidence or higher precedence than Q.
b) Add operators ⊗ to stack.

S6. If right parathesis is encounterd then a) Repeatedly pop from stack & add to P each ⊗ until a left paranthesis is encountered
b) Remove left paranthesis.

S7 Exit.

Consider the following Arithmetic expression Q

Q = 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 *
= ABC * DEF ↑ / G * - H * +

Page No.
Date : / /

• Ex Postfix (Traditional method)

Q. A + (B * C - (D/E↑F) * G) * H
=> A + (B * C - (D/EF↑) * G) * H
=> A + (B * C - (DEF↑/) * G) * H
=> A + ((BC*) - (DEF↑/G*)) * H
=> A + (BC* DEF↑/G* -) * H
=> A + BC* DEF↑/G* -H*
=> ABC* DEF↑/G* -H* +

* Solve using Stack.
Q => A + (B * C - (D/E↑F) * G) * H)

Symbol Scanned	Stack	P
A	)(	A
+	(+	A
(	(+(	A
B	(+(	AB
*	(+(*	AB
C	(+(*	ABC
-	(+(-	ABC*
(	(+(-(	ABC*
D	(+(-(	ABC*D
/	(+(-(/	ABC*D
E	(+(-(/	ABC*DE
↑	(+(-(/↑	ABC*DE
F	(+(-(/↑	ABC*DEF
)	(+(-	ABC*DEF↑/
*	(+(-*	ABC*DEF↑/
G	(+(-*	ABC*DEF↑/G

)	(+	ABC * DEF ↑ / G * -
*	(+ *	ABC * DEF ↑ / G * -
H	(+ *	ABC * DEF ↑ / G * -
)		ABC * DEF ↑ / G * - H

Q. Consider the following Infix expression Q.

Q = ((A+B) * D) ↑ (E - F)

= ((A+B) * D) ↑ (E - F))

Soln:-

Symbol Scanned	stack	P
(	((	
(	(((	
A	(((	A
+	(((+	A
B	(((+	AB
)	((	AB+
*	((*	AB+
D	((*	AB+D
)	(	AB+D*
↑	(↑	AB+D*
(	(↑(	AB+D*
E	(↑(	AB+D*
-	(↑(-	AB+D*E
F	(↑(-	AB+D*E
)	(↑	AB+D*EF-
)		AB+D*EF-↑

Page No.
Date : / /

(Traditional method)

Q. Translate by inspection and hand each Infix expression into equivalent prefix expression.

1) (A-B) * (D/E)
2) (A+B↑D) / (E-F) + G

1] (A-B) * (D/E)
⇒ (-AB) * (/DE)
⇒ *-AB/DE

2] (A+B↑D) / (E-F) + G
⇒ (↑A+BD) / (E-F) + G
⇒ (↑A+BD) / (-EF) + G
⇒ (↑A+BD) /

2] (A+B↑D) / (E-F) + G
⇒ (A+ BD / (-EF) + G

Page No.
Date : / /

Quick Sort

Quick start is based on concept of divide and conquer approach.

Suppose A is following list of 12 numbers as follows

Pivot element ← (44), 33, 11, 55, 77, 90, 40, 60, 99, 22, 88, 66

⇒ Scanned element from right to left
Interchange 22 & 44

22, 33, 11, 55, 77, 90, 40, 60, 99, (44), 88,

Scanned element from left to right
Interchange 44 and 55
22, 33, 11, (44), 77, 90, 40, 60, 99, 55, 88,

Scanned element from Right to left
Interchange 40 and 44
22, 33, 11, 40, 77, 90 (44), 60, 99, 55, 88,

Scanned element from left to right
Interchange 77 and 44
→ 22, 33, 11, 40, (44), 90 77, 60, 99, 55,
66

Page No.
Date : / /

(22), 33, 11, 40
Scanned element from Eight to left
Interchange 22 and 11
21, 33, (22), 40
Scanned element from left to Eight
Interchange 22 and 33
→ 11, (22) 33, 40

(90), 77, 60, 99, 55, 88, 66
Scanned element from Eight to left
Interchange 90 and 66
9 66, 77, 60, 99, 55, 88, (90)
Scanned element from left to Eight
Interchange 90 and 99
66, 77, 60, (90), 55, 88, 99
Scanned element from eight to left
Interchange 90 and 88
→ 66, 77, 60, 88, (95), ~~88~~ (90) 99
Scanned element from left to Eight
~~Interchange~~ 9
→ 66, 77, 60, 88, 55, (90), 99

(66), 77, 60 88, 55
Scanned element from Eight to left

Page No.
Date : / /

Interchange 66 and 55.

55, 77, 60, 88, (66)

Scanned element from left to right

Interchange 66 and 77

55, (66), 60, 88, 77

Scanned element from right to left

Interchange 66 and 66

55, 60, (66), 88, 77

→ 55, 60

→ (88), 77

⇒ 77, 88

∴ ⇒ 11, 22, 33, 40, 44, 55, 60, 77, 88, 90

The complexity of Quick sort is $O(n^2)$

Page No.
Date : / /

* Recursion

- Suppose P is a procedure containing either a call statement to itself or a call statement to a second procedure that may eventually result in a call statement back to the original procedure P. then P is called as recursive procedure.

- The Recursive procedure must have following Two properties

1) There must be a certain criteria called base criteria for which procedure does not call itself

2) Each time the procedure call itself it must be closer to the base criteria

```
Factorial (Fact, N)
① if N=0 then set Fact:= 1
      and Return
② Set Fact := 1
③ Rpeat for k := 1 to N
      set fact := k * Fact
④ Return
```

Page No.
Date : / /

Recursion

Factorial (Fact, N)

1. Copy
2. call factorial (fact, N-1)
3. set fact = N * fact
4. Return

Q. Let a and b denoted as positive integer. Suppose Q is a function define recursively as follows

$$Q(a,b) = \begin{cases} 0 & \text{for } a < b \\ Q(a-b,b)+1 & \text{for } b \leq a \end{cases}$$

1) Q (2, 3)

⇒ Here a = 2, b = 3

Here a < b

Q (2, 3) = 0

2) Q (14, 3)

Here a = 14 and b = 3

Here b ≤ a

Q (14, 3) = Q (a-b, b) + 1

= Q (11, 3) + 1 → a = 11, b = 3

= Q (8, 3) + 1 + 1 ⇒ a = 8, b = 3

= Q (5, 3) + 1 + 1 + 1 ⇒ a = 5, b = 3

Page No.
Date : / /

$= Q(2,3)$ + this means $a=2, b=3$

$= 0 + 4$

$= 4$

Q.2 $Q(J,K) = 5$ for $j < K$

$= Q(J-K, K+2) + J$ for $J \geq K$

1) $Q(2,7)$
2) $Q(5,3)$
3) $Q(15,2)$

① $Q(2,7)$

Here $J=2$ and $K=7$

$\therefore J < K$.

$\therefore Q(2,7) = 5$

② $Q(5,3)$

Here $J=5$, $K=3$

$\therefore J > K$

$\therefore Q(5,3) = Q(2,5) + 35$

Here $j=2$ $K=5$

$j < K$

$\Rightarrow Q(2,5) = 5$

$\therefore 5 + 35$

$= 810$

③

Page No.
Date : / /

③ Q(15, 2)

Here J=15, K=2

j > K

Q(15,2) = Q(13, 4) + 15

j > K

= Q(9, 6) + 15 + 13

= Q(3, 8) + 15 + 13 + 9

= j < K

= 5 + 15 + 13 + 9

= 42.

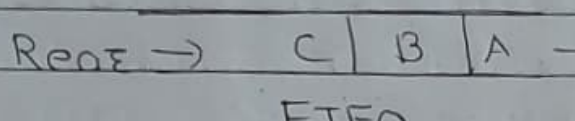
Front Delete
Rear Insert

Page No.
Date : / /

- **Queue [FIFO]**

Queue is a linear list of element in which delation can take place only at one end called 'Front' and inseetion can take place at another end called 'Rear'. Queue are also called as First In First out [FIFO]

e.g A, B, C

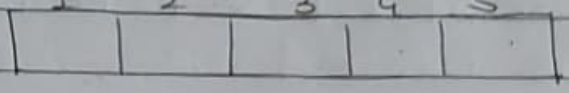


Q. Perform the various operation on Queue Data Structure

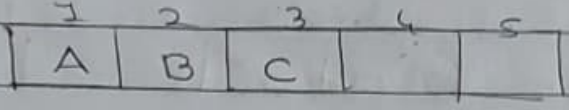
1) Intially empty
2) ABC Inserted
3) A Deleted
4) D and E Inserted
5) B, C Deleted
6) F inserted
7) D Deleted
8) G and H Inserted
9) E Deleted
10) F Deleted
11) K Inserted
12) G and H Deleted
13) K Deleted. Queue empty

Page No.
Date : / /

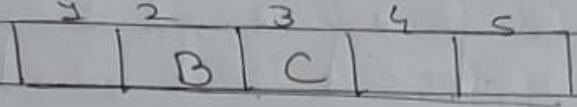
⇒ F = 0
R = 0



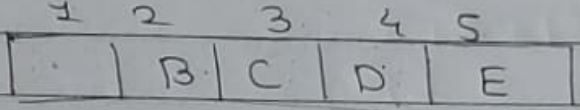
F = 01
R = 3



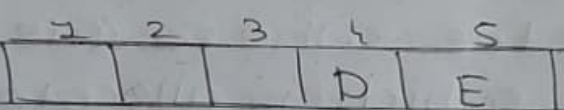
F = 2
R = 3



F = 2
R = 5

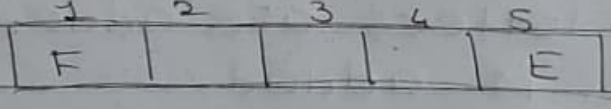


F = 4
R = 5

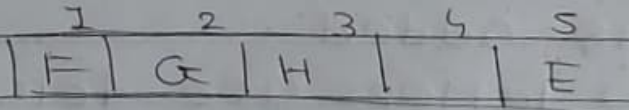


F = 4
R = 1

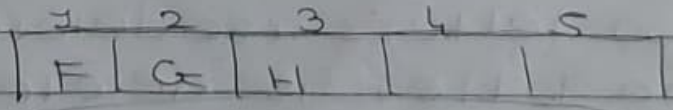
F = 5
R = 1



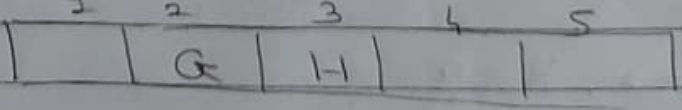
F = 5
R = 3

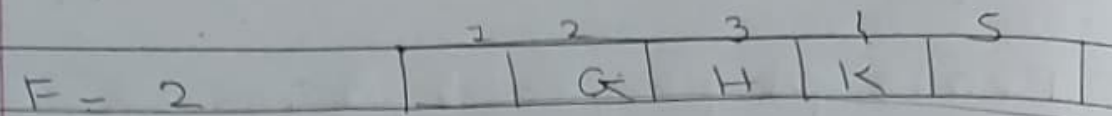


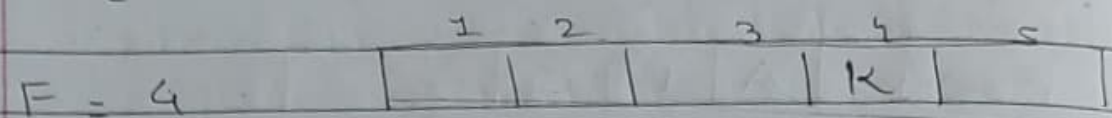
F = 1
R = 3

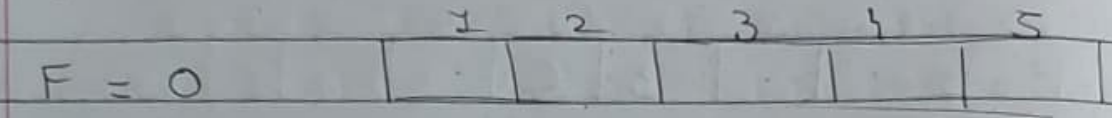


F = 2
R = 3









Q.2. Consider the following queue of characters where Queue is a circular array which is allocated in 6 memory cell

Perform

① F is added
② Two letters are added deleted
③ K, L, M added
④ Two letters are deleted
⑤ R is added
⑥ Two letters are deleted
⑦ T is added
⑧ Two letters are deleted
⑨ one letter is deleted
⑩ One letter is deleted

F = 2, R = 4

—, A, C, D, —,

Page No.
Date : / /

=>

① —, A, C, D, E, — F = 2, R = 5

② —, —, —, D, E, — F = 4, R = 5

③ L, M, —, D, E, K F = 4, R = 2

④ L, M, —, D, E, K F = 6, R = 2

⑤ L, M, R, —, —, K F = 6, R = 3

⑥ —, M, R, —, —, — F = 2 R = 3

⑦ —, M, R, T, —, — F = 2 R = 4

⑧ —, —, —, T, —, — F = 4, R = 4

⑨ —, —, —, —, —, — F = 0, R = 0

⑩ Underflow

Q.3 consider the following Queue allocating with 6 memory cell
F = 2, R = 5

Queue: —, London, Berlin, Rom, Paris, —

Perform :-

① Delhi is added

② Two cities are deleted

Page No.
Date : / /

③ OK is added
④ Mosco is added
⑤ Three cities are deleted
⑥ Bangloore is added

⇒

∴ Algorithim

INSERT (Queue, Front, Rear, item, N)

① If front = 1 and Rear = N
then write overflow & return

② If front := Null (Queue empty)
then Set Front := 1 and Rear := 1

③ if Rear := N then
Set Rear := 1
else set Rear := Rear + 1
④ Set queue [Rear] = item
⑤ Return.

Page No.
Date : / /

: Algorithm :-

DELETE (Queue, Front, Rear, N, item)

① If Front := Null then write := UNDERFLOW and Return.
② Set ITEM := QUEUE [FRONT]
③ if FRONT := Rear
Set Front := Null and
Rear := NULL
Else if Front := N then
set Front := 1
else set Front := Front + 1.
④ Return.

Page No.
Date :

(Pop delete) (Push Insert)
(front) (Rear)

• Dequeue both the operation insertion and deletion performed at both ends

Dequeue is a linear list in which elements can be added or removed at either end but not in the middle. we assume our dequeue is maintain by circular array with pointer left and Right which point to the two ends of Dequeue

e.g consider the following Dequeue of characters where Dequeue is a circular array which is allocated 6 memory cell left = 2 & Eight = 4
Dequeue = —, A, C, D, —, —
Describe the Deque while following operation take place

1) F is added to the Right of Dequeue
2) Two letters on the Right are Delete
3) K, L, M are added to left of Deque
4) One letter on the left is Deleted
5) R is added to the left of Dequeue
6) P is added to the Right of Dequeue
7) T is added to the Right of Dequeue

Page No.
Date : / /

. Evaluate postfix expression with the help of stack

1) 5, 3, +, 2, *, 6, 9, 7, -, /, -
2) 3, 5, +, 6, 4, -, *, 4, 1, -, 2, ↑, +
3) 3, 1, +, 2, ↑, 7, 4, -, 2, *, +, 5, -

1) 5, 3, +, 2, *, 6, 9, 7, -, /, -)

Symbol scanned	Stack
5	5
3	5, 3
+	8
2	8, 2
*	16
6	16, 6
9	16, 6, 9
7	16, 6, 9, 7
-	16, 6, 2
/	16, 3
-	13
)	

Page No.
Date : / /

2] 3, 5, +, 6, 4, −, *, 4, 1, −, 2, ↑, +)

Symbol Scanned	Stack
3	3
5	3, 5
+	8
6	8, 6
4	8, 6 4
−	8, 2
*	16
4	16, 4
1	16, 4, 1
−	16, 3
2	16, 3, 2
↑	16, 9
+	25
)	

3) 3, 1, +, 2, ↑, 7, 4, −, 2, *, +, 5, −)

Symbol Scanned	Stack
3	3
1	3, 1
+	4
2	4, 2
↑	16

Page No.
Date : / /

7	16, 7
4	16, 7, 4
-	16, 3
2	16, 3, 2
*	16, 6
1	22
5	22, 5
-	17
)	