Chapter 2

Stacks and Queues

Contents

ADT Stack and its Operations : Algorithms and their complexity analysis, Applications of Stacks : Expression Conversion and Evaluation - Corresponding algorithms and complexity analysis. ADT queue, Type of Queue : Simple Queue, Circular Queue, Priority Queue; Operations on each types of Queues : Algorithms and their analysis.

POINTS TO REMEMBER

- Stack is a linear data structure in which insertion and deletion of an element can occur only at one end called the top of the stack.
- Push is the term used to insert an element into a stack.
- Pop is the term used to delete an element from a stack.
- Stack can be traversed from top to bottom or bottom to top.
- In stack, elements can be inserted at any place by shifting pointers.
- The element can be deleted by shifting pointer.
- Stacks can be maintained in memory using two methods :
  - (i) Using Array
  - (ii) Using linked list
- If operator symbol is placed between two operands, it is called infix expression.
- If operator symbol is placed before its two operands it is called as prefix expression or polish expression.
- Stack words an the principle of LIFO.
- The stack top or pick operation returns the element at the top of the stack without altering the stack.
- Stacks are used to convert infix expression to post fix expression.
- Stacks are used to sort the elements using quick sort.
- Top pointer contains the location of the top element of the stack.
- A queue is a linear data structure in which new elements are inserted at one end and elements are deleted from the other end.

- A queue follows FIFO (First-in, First-out) property in which elements leave the queue in the order in which they enter.
- The enqueue operation is used to insert a new element to the rear of the queue. The newly inserted element becomes the rear.
- The dequeue operation is used to remove an existing element from the front of the queue.
- The front peek operation returns the element from the front of the queue without altering the queue.
- The rear peek operation is used to return the element from the rear of the queue without altering the queue.
- The circular queue is an improved array representation of a queue. It handles the problem when you want to insert a new element in the queue when rear reaches the end of array and there are empty locations at the beginning of the array representing queue.
- In a circular queue, elements are arranged in such a way that the last element is logically followed by the first element.
- In the linked queue, the actual data item is stored in the INFO part of the node and link part of a node contains a pointer that points to the next element of the queue.
- A deque (double-ended queue) is a linear list that allows insertion and deletion at either end i.e. at the front or rear of the queue but not in the middle.
- Insert left, insert right, delete right and delete left are the four basic operations performed on a deque.
- The deque generalizes both the stack and queue. If you restrict yourself to insert left and delete left operations then deque acts as a stack.
- A priority queue is a variation of simple queue in which each element has been assigned a key called priority. Elements are ordered by priority in the sense that elements with the highest priority are removed first.
- In the linked representation of priority queue, all the nodes are sorted according to the priority values of the elements.
- Queues are used in simulation, some sorting and searching techniques, round robin algorithm etc.

QUESTION-ANSWERS

Q 1. Write procedure to POP top element from STACK. **(PTU, May 2005)**

Ans. POP (STACK, TOP, ITEM)

This procedure deletes the top element of STACK and assigns it to variable ITEM.

1. If TOP = 0, then : Print : Underflow and return :
2. Set ITEM : = STACK [TOP].
3. Set TOP : = TOP – 1
4. Return.

Q 2. What data structure operations can be applied to stacks? (PTU, Dec. 2005)

Ans. Data structure operations that can be applied to stacks are :

1. Push, i.e. inserting an item into stack.
2. Pop, i.e. deleting an item from stack.

These operations are performed only on top of the stack.

Q 3. Write a function to PUSH a new item to STACK. (PTU, Dec. 2005)

Ans. PUSH (STACK, TOP, MAX, ITEM)

1. If TOP = MAX, then :
   Print : overflow & Return.
2. Set TOP = TOP + 1
3. Set STACK [TOP] = ITEM
4. Return.

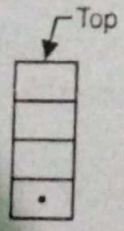
Q 4. What are uses of stacks? (PTU, Dec. 2005)

Ans. 1. Stacks are used to convert infix expression to post fix expression.
2. Stacks are used to evaluate postfix expression.
3. Stacks are used to sort the elements using quicksort.

Q 5. What is stack? (PTU, Dec. 2006 ; May 2006)

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

It is a LIFO system, i.e. last element entered is the first element to delete. Two operations, i.e. push and pop operations are used for insertion and deletion in stacks.



Q 6. Distinguish between stack and queue. (PTU, May 2008, 2007)

Ans.

Stack	Queue
1. As stack is a list of elements in which an element may be inserted or deleted only at one end, called, top of the stack.	1. A queue is a linear list of elements in which deletions can take place only at one end, called, front and insertions can take place at other, called rear end.
2. It is also known as LIFO, i.e. last in first out.	2. It is also known as FIFO, i.e. First In First Out.
3. e.g. Stack of coins are above the other.	3. e.g. Queue of people waiting for a bus.

Q 7. Write any two applications of stack. **(PTU, May 2008)**

OR

List few applications of stack. **(PTU, May 2019, 2009)**

Ans. Application of Stack : The main applications of STACK are as follows :

1. Evaluation of Postfix expression : The stacks are commonly employed in evaluating of postfix expression.

2. Conversion of infix expression to post fix expression : Stack can be used to convert infix expression into the post fix expression.

Q 8. Write the prefix notation for the expression.

(A + B) * C – (D – E) ^ F **(PTU, May 2008)**

Ans. (A + B) *C – (D – E) ^ F

= [+AB] * C – [–DE] ^ F

= [* +ABC] – [^ –DEF]

= –*+ ABC ^ – DEF

Q 9. Define ADT and give an example. **(PTU, May 2009)**

Ans. In computing, an abstract data type or abstract data structure is a mathematical model for a certain class of data structures that have similar behaviour, or for certain data types of one or more programming languages that have similar semantics. An ADT is defined indirectly,only by the operations that may be performed on it and by mathematical constraints on the effects (and possibly cost) of those operations.

For example, an abstract stack data structure could be defined by two operations ; push, that inserts some data item into the structure,and pop, that extracts an item from it;

Example : abstract stack (imperative).

Q 10. Write the postfix notation for the following expression :

(A + B) * C – (D – E) ∧ F **(PTU, Dec. 2008)**

Ans. (A + B) * C – (D – E) ∧ F

The positive notation for the given expression is as follows :

[AB+] * C – [DE–] ^ F

[AB + C*] – [DE – F^]

= AB + C * DE – F^ –

Q 11. What do you understand by polish notation? Explain. **(PTU, May 2010)**

Ans. Polish notation refers to the notation in which the operator symbol is placed before its operands in order from left to right. If an operand is an operaiton with operands, then the same rules apply.

For example :

(X + Y) * Z = [+XY] *Z = * + XYZ

X + (Y * Z) = X + [*YZ] = +X * YZ

Q 12. Give an example that shows how a stack is used by a computer system. **(PTU, May 2010)**

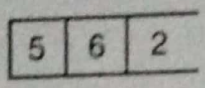
Ans. Stack is internally used by compiler when we execute any recursive function. When a function calls a function, its arguments, return address and local variables are pushed

onto the stack. Since each function runs in its own environment, it becomes possible for a function to call itself. When the function completes its execution these parameters are popped off from the stack to execute the next nested call.

Q 13. Consider the following post fix expression, P : 5, 6, 2, + , *, 12, 4, /, −. Write the procedure and evaluate this expression. **(PTU, Dec. 2004)**

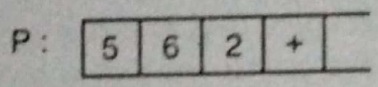
Ans.

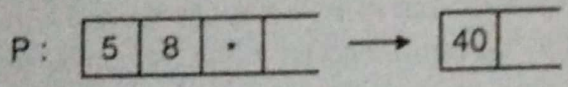
1. Add a right parenthesis at the end of post fix expressions.
   P : 5, 6, 2, +, *, 12, 4, /, −)
2. Scan P from left to right and repeat the following steps for each element of 'P' until right parenthesis is encountered.
3. If an operand is encountered, put it on the stack.

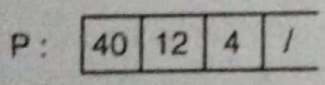


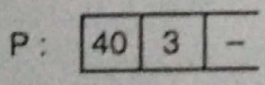
4. If an operator is encountered, then, remove the top 2 elements, apply operator and place the result back on stack.

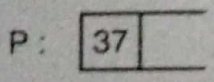
∴











∴ Value = 37

Symbols scanned	Stack
5	5
6	5, 6
2	5, 6, 2
+	5, 8
*	40
12	40, 12
4	40, 12, 4
/	40, 3
−	40, 3
)	37

Q 14. What is a top pointer of a stack? **(PTU, Dec. 2010)**

Ans. Top pointer which contains the location of the top element of the stack ; and a variable MAXSTK which gives the maximum number of elements that can be held by the stack. The condition TOP = 0 or TOP = NULL will indicate that the stack is empty.

Q 15. Consider the postfix expression :
P : 12, 7, 3, –, /, 2, 1, 5, +, *, +.
Write the procedure and evaluate the expression. **(PTU, Dec. 2005)**

Ans.

1. Add a right parenthesis at the end of post fix expression.
2. Scan P from left to right and repeat to following steps for each element of 'P' until right parenthesis is encountered.
3. If an operand is encountered, put it on stack.
4. If an operator is encountered, then remove the top two elements, apply operator and place the result back on stack.

Symbol	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
)	15

∴ Value = 15.

Q 16. Consider the following infix expression :
((A + B) * D) ↑ (E – F).
Write the expression and convert into the equivalent postfix expression.
(PTU, May 2005)

Ans.

Symbol	Stack	P
(	((	
(	(((	
A	(((	A
+	(((+	A
B	(((+	AB

)	((	AB +
*	((*	AB +
D	((*	AB + D
)	(	AB + D*
↑	(↑	AB + D*
(	(↑(	AB + D*
E	(↑(	AB + D*E
–	(↑(–	AB + D*E
F	(↑(–	AB + D*EF
)	(↑	AB + D* EF –
)		AB + D* EF – ↑

Hence, postfix expression.

P : AB + D * EF – ↑

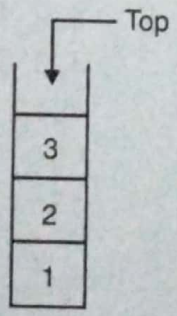
Q 17. Explain concept of stack along with its applications. (PTU, Dec. 2006)

Ans. A stack is a list of elements in which an element may be inserted or deleted only at one end, called top of the stack. Here, the elements are removed from a stack in reverse order of that in which they were inserted into the stack. Stack is also called a Last-in First-out (LIFO) structure.

Two basic operations associated with stacks :

(i) "Push" is used to insert an element into stack.

(ii) "Pop" is the term used to delete an element from a stack.



Application of stacks :

1. Evaluation of postfix expressions : Stacks are commonly employed in evaluation of postfix expression P. Various steps involved are :

(i) Add a right parenthesis at the end of postfix expression.

(ii) Scan 'P' from left to right and repeat the following steps for each element until right parenthesis is encountered.

(iii) If an operand is encountered, put it on stack.

(iv) If an operator is encountered, then remove the top two elements, apply the operator and place the result back on stack.

2. Conversion of infix expresssion to postfix : Stacks are also used to convert an infix expression to postfix expression.

e.g. If an infix expression is :

5* (6 + 2) – 12/4

Symbol	Stack	P
5	(	5
*	(*	5
6	(* (	56
+	(* (+	562
2	(* (+	562
)	(*	562+
–	(–	562 + *
12	(–	562 + *12
/	(–/	562 + * 12
4	(–/	562 + * 124
)		562 + * 124/-

∴ Equivalent postfix expression is

562 + * 124/-

3. Quicksort : Quicksort is an algorithm which uses divide and conquer technique to sort the elements. Stacks are used for sorting. In this algo, we position the first element by applying some iteration at its correct position and divide the complete list in two parts and by using stack, we keep track of these two lists.

Q 18. Write a program for implement stack using arrays. **(PTU, May 2006)**

Ans.

```
#include <iostream.h>
#include <process.h>
void push (int a [ ], int n) ;
void pop (int a [ ] );
void display (int a [ ], int) ;
int top = –1;
void main ( )
{
int a [51], n ;
char ch = 'y';
while (ch = = 'y' || ch == 'Y')
{
cout << "Insert element into stack",
cin >> n ;
push (a, n) ;
cout << "Want to insert more?" ;
cin >> ch ;
```

```
}
do
{
cout <<"Current stack is" ;
display (a, top) ;
cout <<"want to delete item?" ;
cin >> ch;
if (ch = = 'y' || ch = = 'Y')
pop (a) ;
else
ch = 'n' ;
} while (ch = = 'y' || ch = 'Y') ;
cout << "stack is" ;
display (a, top) ;
}
void push (int a [ ], int n)
{
if (top = = 51)
{
cout <<"overflow" ;
exit (0) ;
}
top = top + 1 ;
a [top] = n ;
}
void pop (int a [ ])
{   if (top = = -1)
    {   cout <<"under flow" ;
        exit (0) ;
    }
    int them = a [top] ;
    cout <<" Item deleted : "<< item ;
    top = top - 1 ;
}
void display (int c [ ] , int top)
{
for (int i = top ; i > = 0 ; i - -)
cout <<" "<< a [i] ;
}
```

Q 19. What is the polish notation representation of the following expression?

(A * (b + C) + (b/d)* a + z

(a + (b + c* (d + e))) + f **(PTU, May 2010)**

Ans.

1. (A * (b + C)) + (b/d) * a + z
   ≈ (A* [+bC]) + [/bd] * a + z
   ≈ [* A + bC] + [*/bda] + z
   ≈ [+*A + bC*/bd a] + Z
   ≈ ++ *A + bC*/bdaz

Hence, the expression in polish notation is :

++*A + bC */ bdaz

2. (a + (b + C* (d + e))) + f
   ≈ (a + (b + C* [+de])) + f
   ≈ (a + (b + [*C + de])) + f
   ≈ (a + [+b*c + de]) + f
   ≈ [+a + b * c + de] + f
   ≈ ++a + b *c + def

Hence, the expression in polish notation is :

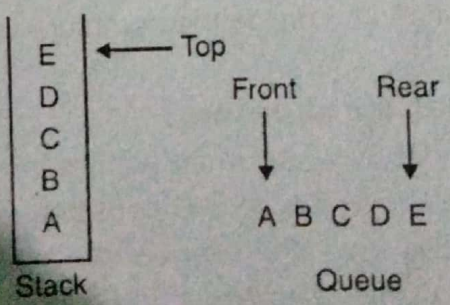
++ a + b*c + def

Q 20. Compare stacks and queues and discuss their limitations.

(PTU, Dec. 2007)

Ans. Two of the more common data objects found in computer algorithms are stacks and queues. Both of these objects are special cases of the more general data object, an ordered list.

A stack is an odered list in which all insertions and deletions are made at one end, called the top. A queue is an ordered list in which all insertions take place at one end, the rear, while all deletions take place at the other end, the front. Given a stack S = (a [1], a [2], ... a [n]) then we say that a1 is the bottom most element and element a [i]) is on top of element a [i – 1], 1 < i < = n. When viewed as a queue with a [n] as the rear element one says that a [i + 1] is behind a [i], 1 < i < =n.



The restrictions on a stack imply that if the elements A, B, C, D, E are added to the stack, then the first element to be removed/deleted must be E. Equivalently we say that the last element to be inserted into the stack will be the first to be removed. For this reason stacks are sometimes referred to as Last In First Out (LIFO) lists. The restrictions on queue imply that the first element which is inserted into the queue will be the first one to be removed. Thus A is the first letter to be removed, and queues are known as First In First Out (FIFO) lists. Note that the data object queue as defined here need not necessarily correspond to the mathematical concept of queue in which the insert/delete rules may be different.

Q 21. What is the post fix and prefix representation of the following expression (A * (b + C) + (b/d)* a + z **(PTU, Dec. 2010)**

Ans. Step 1. Fully parenthesize the expression,

I = ((A * (b + c)) + ((b/d) * (a + z)))

Step 2. Convert to postfix expression.

I = ((A * (bc+)) + ((b/d) * (a + z)))
I = ((A * (bc+)) + ((bd/) * (a + z)))
I = ((A * (bc+)) + ((bd/) * (az+)))
I = ((A (bc+*) + ((bd/) * (az+)))
I = ((A (bc+)*) + ((bd/) (az +)*))
I = ((A (bc+)*) ((bd/) (az+)*) +)

Q 22. Give application of stacks, queues and trees. Also list the operations possible on stack. **(PTU, May 2004)**

Ans. The application of stacks are as follow :

1. Evaluation of post fix expression :

Stacks are commonly employed in evaluation of a postfix expression.

2. Conversion of infix expression to postfix expression :

Stacks are used to convert an infix expression to postfix expression.

e.g. It infix expression is :

5* (6 + 2) – 12/4

Then, using stacks, the postfix expression is :

562 + *124/-

3. Quick sort : Quicksort is an algorithm which uses divide and conquer technique to sort the elements. Stacks are used for sorting. Here, we position the Ist element by applying some iteration at its correct position and divide the complete list in 2 parts and by using stack, we keep track of these 2 lists.

The application of queues are as follow :

1. Breadth First Search (BFS) : The main application of queues is in traversing the path using Breadth First Search (BFS), i.e. to find path between and 2 vertices in graph.

2. Priority queues : Priority queues are used to assign priority, s.t. an element of higher priority is processed before any element of lower priority.

The application of trees are as follow :

1. Trees are used to find the path length using Huffman; algorithm

$$L_E = L_I + 2^*n$$

Where, $n \rightarrow$ no. of internal nodes

$L_E \rightarrow$ length of external nodes

$L_J \rightarrow$ length of internal nodes.

2. Huffman's algorithm is used to find a tree with minimum weighted path. Operations possible on stack are :

1. **Push :** PUSH operation is used to insert an element at the top of the stack.

PUSH (STACK, TOP, MAX, ITEM)

1. If TOP = MAX, then :
   Print : overflow and return
2. Set TOP = TOP + 1
3. Set STACK [TOP] = ITEM
4. Return.

2. **POP :** Pop operation is used to delete an item for Top of the stack.

POP (STACK, TOP)

1. If TOP = NULL, then :
   write : underflow and Exit
2. Set ITEM = STACK [TOP]
3. Set TOP = TOP – 1
4. Exit.

Q 23. Consider the following mathematical expression written in postfix :
12, 9, 3, –/, 3, 2, 5, +, *, +.
Write the procedure and evaluate the expression. **(PTU, May 2005)**

Ans.

Symbols scanned	Stack
12	12
9	12, 9
3	12, 9, 3
–	12, 6
/	2
3	2, 3
2	2, 3, 2
5	2, 3, 2, 5
+	2, 3, 7
*	2, 21
+	23
)	

∴ Value = 23.

Q 24. Explain about infix, prefix and postfix and write an algorithm to convert infix to postfix expression. **(PTU, May 2006)**

Ans. For most common arithmetic operation, operator symbol is placed between its two operands. For e.g. A + B, C – D, E * F etc.

This is called infix notation.

(i) Polish notation, i.e. prefix notation refers to notation in which the operator symbol is placed before its two operands. **e.g.**
+AB, –CD, *EF etc.

(ii) Reverse polish notation, i.e., postfix expression is an analogous notation in which operator symbol is placed after its two operands. **e.g.**
AB+, CD–, EF*, etc.

Polish (Q, P)

Suppose 'Q' is an arithmetic expression written in infix notation. This algorithm finds the equivalent postfix expression P.

1. Push "("onto STACK, and add")" to end of Q.
2. Scan 'Q' from left to right and repeat steps 3 to 6 for each element of 'Q' until STACK is empty.
3. If an operand is encountered, add it to P.
4. If a left parenthesis is encountered, push it onto stack.
5. If an operator (X) is encountered, then (a) Repeatedly pop from STACK and add to P each operator which has same precedence or higher precedence then X (b) Add X to STACK.
6. If a right parenthesis is encountered, then :
   (a) Repeatedly pop from STACK and add to 'P' each operator until a left parenthesis is encountered.
   (b) Remove the left parenthesis.
7. Exit.

Q 25. What are the various operations possible on stacks? Explain the algorithm for each of them. **(PTU, May 2019 ; Dec. 2010)**

Ans. Stacks : A stack is a list of elements in which an element may be inserted or deleted only at one end, called the top of the stack. This means, in particular, that elements are removed from a stack in the reverse order of that in which they were inserted into the stack.

Special terminology is used for two basic operations associated with stacks :

(a) "Push" is the term used to insert an element into a stack.

(b) "Pop" is the term used to delete an element from a stack.

PUSH (STACK, TOP, MAXSTK, ITEM)

This procedure pushes an ITEM onto a stack.

1. [stack already filled?]
   If TOP = MAXSTK, then : Print : OVERFLOW and Return.

2. Set TOP = TOP + 1 [Increases Top by 1]
3. Set STACK (TOP) = ITEM [Inserts ITEM in new TOP position]
4. Return.

POP (STACK, TOP, ITEM)

This procedure deletes the top element of STACK and assigns it to the variable ITEM.

1. [Stack has an item to be removed?]
   If TOP = 0, then : Print : UNDERFLOW, and Return.
2. Set ITEM = STACK [TOP]. [Assigns TOP element to ITEM]
3. Set TOP = TOP – 1 . [Decreases Top by 1]
4. Return.

Push an Item into Stack :

```
/ * C implementation of pop algorithm *.
int pop (int stack [ ], int stack – top) / * stack – top is a local variable indicating top of
stack * /
{int item ;
   if (stack_top <10)
   {   print f ("\ n stack underflow") ;
       return – 1 ;
   }
  else
   {   item = stack [stack – top] ;
       stack – top ..... ;
       top = stack – top ; / * update global variable
       top * /
       return item ;
   }
}
```

POP from stack :

```
/ * C implementation of push algorithm * /
int push (int stack [ ], int stack – top, int maxstack, int item)
{   if (stack – top = = maxstack)
    {   print f (" \n stack overflow") ;
            return – 1 ;
    }
   else   {   stack – top ++ ;
              stack [stack – top] = item ;
              top = stack – top ; /* update global variable top* /
              return 0 ;
          }
}
```

Q 26. Convert the following infix expression into prefix and postfix notations.
((a + b) + c * (d + e) + f) * (g + h). **(PTU, May 2009)**

Ans. The postfix expression is :

((a + b) + c * (d + e) + f) * (g + h)
= ((ab+) + C * (de+) + f) * (gh+)
= ((ab+) + (cde + *) + f) * (gh+)
= ((ab + cde + * +) + f) * (gh+)
= (ab + cde + * + f +) * (gh+)
= (ab + cde + * + f + gh + *)

The prefix expression is :

((a + b) + c * (d + e) + f) * (g + h)
= ((+ab) + c * (+de) + f) * (+gh)
= ((+ab) + (* c + de) + f) * (+gh)
= ((++ ab * c + de) + f) * (+gh)
= (+++ ab * c + def) * (+gh)
= (* +++ ab * c + def + gh)

Q 27. Write an algorithm to convert infix expression to postfix expression. Give one example and apply algorithm. **(PTU, May 2007)**

Ans. POLISH (Q, P)

1. Push "("onto STACK & add")" to end of Q.
2. Scan 'Q' from left to right and repeat steps 3 to 6 for each element of until STACK is empty.
3. If an operand is encountered, add it (X) to P.
4. If a left parenthesis is encountered, push it onto stack.
5. If an operator (X) is encountered, then :
   (i) Repeatedly pop from STACK and add to P each operator which has same or higher procedence than (X)
   (ii) Add (X) to stack.
6. If a right parenthesis is encountered, then :
   (i) Repeatedly pop from STACK and add to P each operator until a left parenthesis is encountered.
   (ii) Remove the left parenthesis.
7. Exit.

e.g. Let Infix expression be :
((A + B) *D) ↑ (E – F)

Symbol	Stack	P
(	((	
(	(((	
A	(((	A
+	(((+	A
B	(((+	AB
)	((	AB+
*	((*	AB+
D	((*	AB+D
)	(	AB+D*
↑	(↑	AB+D*
(	(↑ (	AB+D*
E	(↑ (	AB + D*E
–	(↑(–	AB+D*E
F	(↑(–	AB + D* EF
)	(↑	AB + D*EF –
)		AB + D*EF–↑

∴ Post fix expressison is :

AB + D * EF – ↑

Q 28. Convert the following infix expression to postfix.

A + (B * C – (D / E ↑ F) * G) *H **(PTU, Dec. 2007)**

Ans. Q : A + (B * C – (D / E ↑ F) * G) *H

A + (B * C – (D / E ↑ F) * G) * H)

(1) (2) (3) (4) (5) (6) (7) (8) (9) (10) (11) (12) (13) (14) (15) (16) (17) (18) (19) (20)

The elements of Q have now been labelled from left to right for easy reference. The diagram shows status of STACK and of the string P as each element of Q is scanned.

1. Each operand is simply added to P and does not change STACK.
2. The subtraction operator (–) in row 7 sends * from STACK to P before it (–) is pushed onto STACK.
3. The right paranthesis in row 14 sends ↑ and then / from STACK to P, and then removes the left parenthesis from TOP of the STACK.
4. The right paranthesis in row 20 sends * and thus + from STACK to P and then removed left paranthesis from TOP of the STACK.

After step 20 is executed, the STACK is empty one

P : ABC * DEF ↑ / G * – H * +

which is requierd postfix equivalent of Q.

Symbol scanned	STACK	Expression P
(1) A	(	A
(2) +	(+	A
(3) (	(+ (	A B
(4) B	(+ (*	A B C
(5) *	(+ (*	A B C *
(6) C	(+ (–	A B C *
(7) –	(+ (– (	A B C * D
(8) (	(+ (– (	A B C * D
(9) D	(+ (– (	A B C * D E
(10) /	(+ (– (/	A B C * D E
(11) E	(+ (– (/	A B C * D E F
(12) ↑	(+ (– (/ ↑	A B C * D E F ↑ /
(13) F	(+ (– (/ ↑	A B C * D E F ↑ /
(14))	(+ (–	A B C * D E F ↑ / G
(15) *	(+ (– *	A B C * D E F ↑ / G * –
(16) G	(+ (– *	A B C * D E F ↑ / G * –
(17))	(+	A B C * D E F ↑ / G * – H
(18) *	(+ *	A B C * D E F ↑ / G * – H
(19) H	(+ *	* +
(20))		

Q 29. Write the postfix notation for the following expression :
(A + B) * C – (D – E) ^ F. **(PTU, May 2011)**

Ans. Postfix notation of (A + B) * C – (D – E) AF is equal to AB + C* DE – FA

Q 30. Write any two applications of stack. **(PTU, May 2011)**

Ans. The two applications of stack are :

1. Towers of hanoi
2. Recursion
3. Conversion of expression.

Q 31. Write down the algorithms for various operations possible on stacks. **(PTU, May 2011)**

Ans. Stacks : Stacks is a linear structure which consist of list of data items in which insertions and deletion are made only at one end called the top of the stack. Also, arc the data item or elements can be removed or added only from the top, it is defined as LAST-IN-FIRST-OUT (LIFO) structure. The terms used in operation associated with stack are PUSH and POP.

1. Push : Push is used to insert an element into a stack.

2. Pop : Pop is used to delete an element from a stack.

The stack structure functions in the same manner as a spring loaded stack of plates or tray function in canteen. When we add plates by pushing the top of the stack and to remove a plate from the top of the stack the spring causes the next plate to pop up.

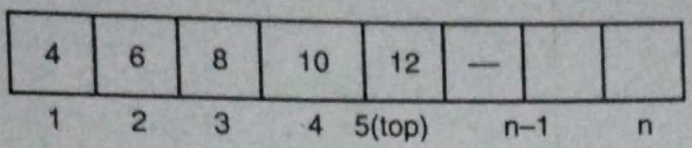
Representation of Stacks :

e.g. If the following 4 elements are pushed into empty stack
, 6, 8, 10, 12

then the stack is written as : 4, 6, 8, 10, 12

where right most element is 12 which is at the top of the stack.

It shown is the following fig.



Algorithm :

Step 1. If top = MAX
Print "Overflow" and return

Step 2. Set top = top + 1 Shows increment of value.

Step 3. Set stack (top) = D It adds data in the new top position

Step 4. Return

Algorithm for the pop (Deletion of element from the STACK)

Step 1. If top = 0
Print "Underflow" and return – It shows that stack has no value to remove

Step 2. Set D = Stack (Pop) It assign the top element of the stack to DATA

Step 3. Set top = top = 1 Increases the value of pointer variable top.

Step 4. Return.

Q 32. What are the various operations an stack?

Ans. There are two operations can be applied an stack :

(a) PUSH operation

(b) POP operation

(a) Push Operation : Push means to insert a new item at top of stack. In executing the procedure PUSH, we must first check whether there is a space in stack to insert new item or not. If there is a space then we insert new element otherwise the condition of overflow occurs.

(b) Pop Operation : Pop means to delete the top element from stack. In Pop operation, one must first test whether there is an element in the stack to be deleted, if not then we have the condition known as underflow.

Q 33. Convert the infix notation 1 + (2*3 – (4/5 ↑ 6)*7) *8 into postfix notation.
(PTU, Dec. 2011)

Ans. Q is the infix notation and transform it into its equivalent postfix expression P.

First we push "(" onto stack, and then we add ")" to the end of Q to obtain.

Q : 1 + (2*3 – (4/5 ↑ 6)*7) *8)

(1) (2) (3) (4) (5) (6) (7) (8) (9) (10) (11) (12) (13) (14) (15) (16) (17) (18) (19) (20)

The elements of Q have now labeled from left to right. Table shows the status of stack and of the string P as each element of Q is scanned.

After step 20 is executed, the stack is empty and.

P : 123* 456↑ /7 * – 8* +

Which is the required postfix equivalent of Q.

	Symbol Scanned	Stack	Expression 'P'
(1)	1	(	1
(2)	+	(+	1
(3)	(	(+ (	12
(4)	2	(+ (	12
(5)	*	(+ (*	123
(6)	3	(+ (*	123*
(7)	–	(+ (–	123*
(8)	(	(+ (–(	123*4
(9)	4	(+ (– (	123*4
(10)	/	(+ (– (/	123 * 45
(11)	5	(+ (– (/	123*45
(12)	↑	(+ (– (/ ↑	123*456
(13)	6	(+ (– (/ ↑	123*456↑/
(14)	)	(+ (–	123*456↑/
(15)	*	(+ (– *	123*456 ↑/7
(16)	7	(+ (– *	123*456↑ 7*–
(17)	)	(+	123*456 ↑ /7* –
(18)	*	(+ *	123*456↑/7*–8
(19)	8	(+ *	123*456↑/7*–8*+
(20)	)	*	

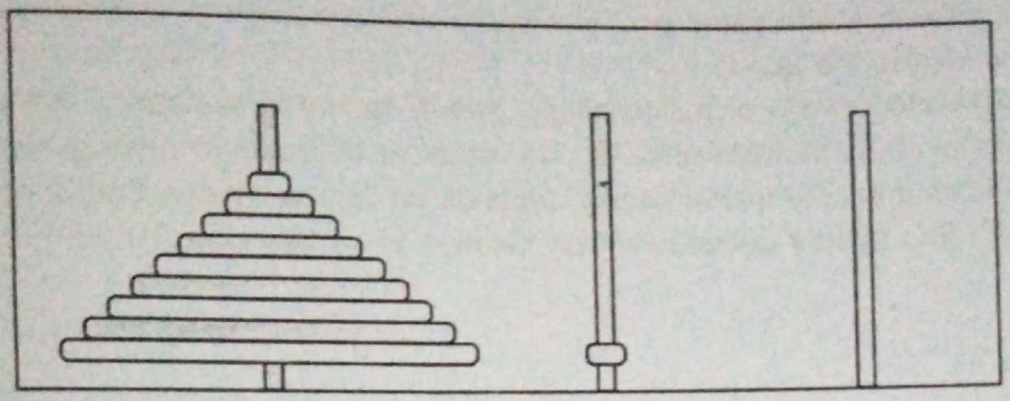
Q 34. Briefly explain Tower of Hanoi problem. **(PTU, Dec. 2011)**

Ans. The Tower of Hanoi puzzle was invented by the French mathematician Edouard Lucas in 1883. We are given a tower of eight disks (initially four in the applet below), initially stacked in increasing size on one of three pegs. The objective is to transfer the entire tower to one of the other pegs (the rightmost one in the applet below), moving only one disk at a time and never a larger one onto a smaller.

The puzzle is well known to students of Computer Science since it appears in virtually any introductory text on data structures or algorithms. Its solution touches on two important topics :

- ❑ Recursive functions and stacks
- ❑ Recurrence relations

The applet has several controls that allow one to select the number of disks and observe the soluton in a Fast or Slow manner. To solve the puzzle drag disks fron one peg to another following the rules. You can drop a disk on to a peg when its center is sufficiently close to the center of the peg. The applet expects you to move disks from the leftmost peg to the rightmost peg.



Q 35. List types of queues. **(PTU, May 2004)**

Ans. The queues are of the types : simple queue, circular queue, dequeue and priority queues.

Q 36. A deque is maintained by a circular array with N memory cells. When an element is added to deque, how is LEFTand RIGHT 2 variables to indicate 2 ends of a deque changed? **(PTU, Dec. 2004)**

Ans. If the element is added on the left, then LEFT is 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).

Q 37. What are limitations of queues? **(PTU, Dec. 2005)**

Ans. 1. The major drawbacks is that one cannot have access to middle element, i.e. insertion and deletion can take place only at the two ends called front end and rear end.

2. In this case, last element in a queue will be last element out of queue i.e. if we want to accese only last element, we have to transverse all the elements before it.

Q 38. What is priority queue. **(PTU, Dec. 2009 ; May 2014, 2006)**

Ans. Priority queue is the collection of elements such that each element has been assigned a priority and the order in which the elements are deleted and processed comes from following rules :

1. An element of higher priority is processed before any element of lower priority.
2. Two elements with same priority are processes according to order in which they were added to queue.

Q 39. Write down application of queues. **(PTU, May 2007)**

Ans. 1. The main application of queues is that it is used in traversing the graph using Breadth First Search (BFS) i.e. to find shortest path between two vertices.

2. Priority queues are used to assign priority s.t., an element of higher priority is processed before any element of lower priority.

Q 40. What are priority queues? How they are implemented? **(PTU, Dec. 2007)**

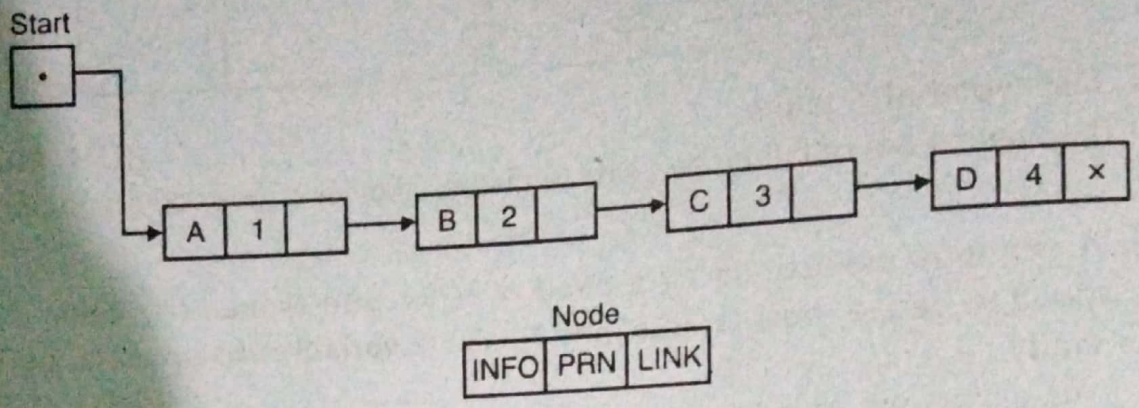
Ans. Priority Queues : A priority queue is a collection of elements such that the order in which elements are deleted and processed comes from the following rules :

1. An element of higher priority is processed before any element of lower priority.

2. Two elements with same priority are processed according to order in which they are added to the queue.

There are various ways of maintaining a priority queue in memory : one uses a one-way list and other uses multiple queues. The ease or difficulty in adding elements to or deleting them from a priority queue clearly depends on representation that one choosen.

Example : The priority queue is shown below :



Priority queue with 4 node

Q 41. What is a queue? Discuss operations on queue. **(PTU, Dec. 2007)**

Ans. Queue : A queue is a linear data structure in which new elements are inserted at one end and elements are deleted from the other end.

Operation on Queue : A number of operation can be performed on a queue. These functions are :

Enqueue : Used to insert a new element to rear of queue.

Dequeue : Used to remove an existing element to front of queue.

Create : Used to create an empty queue.

Destroy : Used to delete all the element of the queue.

Isempty : Check and report is queue empty or not.

Isfull : Check and report is queue full or not.

Also perform front peek and rear peek.

Q 42. What are Dequeues? How are they maintained in memory? (PTU, May 2005)

Ans. Dequeue is a linear list in which elements can be added or removed at either end, but not in the middle.

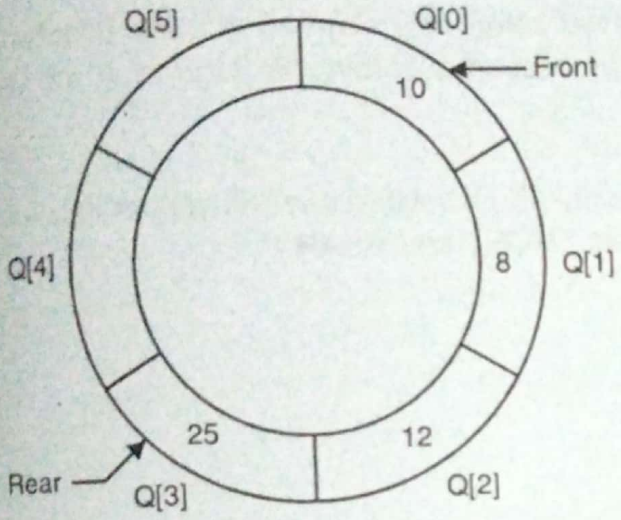
Dequeues is maintained by a circular queue with pointers 'LEFT' and 'RIGHT', which points to 2 ends of a dequeue. The variations of dequeue are :

(i) I/P restricted dequeue : In this type, insertion can only be possible at one end, but deletion can be done on both ends.

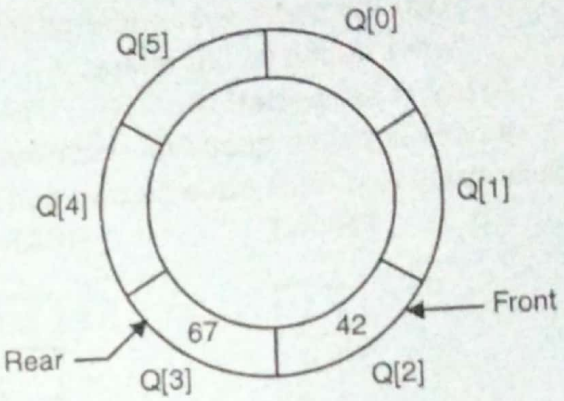
(ii) O/P restricted dequeue : In this type, deletion is possible only at one end, but insertion can be done at both ends.

Q 43. What is a circular queue? **(PTU, May 2010)**

Ans. In circular queues the elements are represented in a circular fashion so that last node points back to the first node. In circular queue, the insertion of a new node or element is done at the very first location of the queue if the last location at the queue is full. Suppose Q is a queue array of 6 elements. Insertion and deletion operation can be performed on circular queue. The following figures illustrate the insertion and deletion operations.



A circular queue after inserting 10, 8, 12, 25



A circular queue after deletion of 10, 8

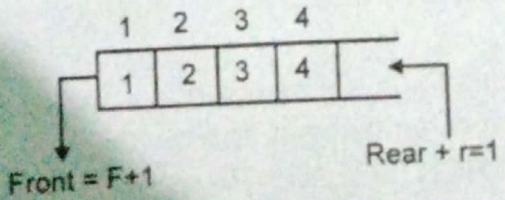
Q 44. What are queues? Compare with dequeues. How are they represented in memory? **(PTU, Dec. 2004)**

OR

How queues are represented in memory? **(PTU, May 2019 ; Dec. 2008)**

Ans. A queue is a linear list of elements in which deletions can take place only at one end called the front and insertion can take place only at the other end called the rear end.

Queues are also called FIFO, since Ist element in a queue will be Ist element out of queue.



With every insertion, rear is incremented by 1. With every deletion, front is incremented by 1.

Dequeues : It is a linear list in which elements can be added or removed at either end, but not in the middle. Dequeue is maintained by circular array with pointers 'LEFT' and 'RIGHT' which points to the two ends of a deque. The variations of dequeue are :

(i) Input restricted dequeues : In this type, insertion can only be possible at one end but deletion can be done on both the ends.

(ii) O/P restricted dequeues : In this type of dequeues, deletion is possible only at one end, but insertion can be done at both the ends.

Q 45. What are priority queues? Explain how are priority queues represented by using arrays? **(PTU, Dec. 2005)**

Ans. Priority queue is the collection of elements such that each element has been assigned a priority and the order in which the elements are deleted and processed comes from following rules :

1. An element of higher priority is processed before any element of lower priority.
2. Two elements with same priority are processed according to order in which they were added to the queue.

Array representation

It uses separate queue for each level of priority. Each such queue will appear in its own circular array and must have its own pair of points FRONT and REAR.

e.g. **FRONT** **REAR**

	FRONT
1	1
2	4
3	2
4	0
5	6

	REAR
1	3
2	6
3	2
4	0
5	1

	1	2	3	4	5
1	A	B	C		
2				D	E
3		G			
4					
5	K				H

Q 46. Write suitable routines to perform insertion and deletion operations in a queue. **(PTU, May 2009)**

Ans. The operations of insertion and deletion in a queue as follows :

An algorithm for insertion of an element in a queue.

Q INSERT(QUEUE, N, FRONT, REAR, ITEM)

Step 1: [Queue alread filled?]
If FRONT = 1 and REAR = N or if FRONT = REAR + 1, then :
Write : OVERFLOW and Return

Step 2 : [Find new value of REAR]
If FRONT = NULL then [queue initially empty]
Set FRONT = 1 and REAR = 1

Else if REAR = N then
Set REAR = 1
Else
Set REAR = REAR +1
[End of If data structure]

Step 3 : Set QUEUE [REAR] = ITEM This insert new line

Step 4 : Return.

An algorithm for deletion from queue

QUEUE (QUEUE, N, FRONT, REAR, ITEM)

Step 1: [Queue already empty 1]
If FRONT = NULL the write UNDERFLOW and Return.

Step 2 : Set ITEM = QUEUE [FRONT]

Step 3 : [Find new value of FRONT]
If FRONT = REAR then
Set FRONT = NULL and REAR = NULL
Else if FRONT = N then
set FRONT = 1
Else
Set FRONT = FRONT + 1
[End of If structure]

Step 4 : Return.

Q 47. What are the various operations possible on queue? Explain the algorithm for each of them. **(PTU, May 2010)**

Ans. A queue is logically a first-in first-out (FIFO) linear data structure. It is a homogeneous collection of elements in which new elements are called at one end called rear, and the existing elements are deleted from other end called front.

The basic operations that can be performed on queue are :

1. Insert (add) an element to the queue.
2. Delete (remove) an element from a queue.

Let Q be the array of some specified size say size.

(a) Following in an algorithm for inserting an element into the queue :

1. Initialize front = 0, rear = –1
2. Input the value to be inserted and assign to variable "data".
3. If (Rear > = SIZE)
   (a) Display "Queue overflow"
   (b) Exit.
4. Else
   (a) Rear = Rear + 1
5. Q [Rear] = data
6. Exit

(b) Following is an algorithm for deleting an element from queue

1. If (Rear < Front)
   (a) Front = 0, rear = −1
   (b) Display "The queue is empty"
   (c) Exit.
2. Else
   Data = Q [Front]
3. Front = Front + 1
4. Exit.

Q 48. Explain the procedure for insertion of an item into a queue. **(PTU, May 2005)**

Ans. Suppose we want to insert an element ITEM into a queue at the time the queue does occupy the last part of the array, i.e. when REAR = N.

As the array QUEUE is circular, i.e. QUEUE [i] comes after QUEUE [N] in aray with this assumption, we insert ITEM into queue by assigning ITEM to QUEUE [1]. Thus, instead of increasing REAR to N + 1, we reset REAR = 1

QUEUE [REAR] = ITEM

Now, whenever an element is added to queue, the value of REAR is increased by 1, i.e.

REAR = REAR + 1

The condition FRONT = NULL indicates that queues is empty.

INSQUEUE (FRONT, REAR, QUEUE)

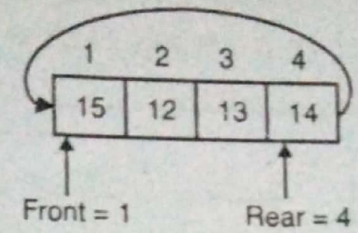
1. If FRONT = 1, REAR = N, then :
   Write : overflow & Exit.
2. If FRONT = REAR = NULL
   Set FRONT = REAR = 1
   ELSE IF REAR = N & FRONT ≠ 1,
   Set REAR = 1
   ELSE
   Set REAR = REAR + 1
3. Set QUEUE [REAR] = ITEM
4. Exit.

Q 49. Explain various types of queues with examples and write an algorithm to implement circular queue. **(PTU, May 2006)**

Ans. A queue is a linear list of elements in which deletions can take place only at one end called front and insertion can take places only at other end called rear end.

Types of queues :

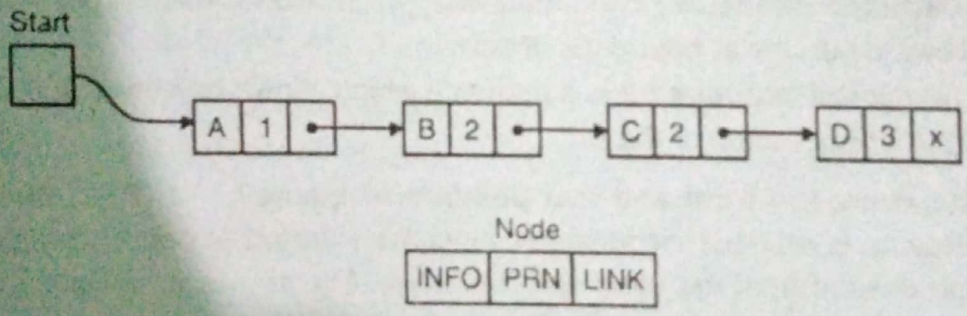
1. Circular queue : These are used to better utilize the memory location. In this case, Q [1] comes after Q [N], i.e. last element. If FRONT = N and element is deleted, we reset FRONT = 1 instead of increasing FRONT = N + 1



If Front = Null and Rear = Null, their, queue is empty. Suppose we have only 1 element, then, Front = Rear ≠ Null.

(ii) Priority Queues : Priority queue is the collection of elements s.t. each element has been assigned a priority and the order in which elements are deleted and processed comes from following rules.

1. An element of higher priority is processed before any element of lower priority.
2. Two elements with same priority are processed according to order in which they were added to queue.



A node 'X' preceeds a node 'Y' when 'X' has priority greater than 'Y' or both have same priority, but 'X' was added to list before 'Y'.

3. Dequeues : It is a linear list in which elements can be added or removed at either end, but not in the middle. Dequeue is maintained by a circular queue with pointers 'LEFT' and 'RIGHT' which points to two ends of dequeue. These are of two types :

1. I/P restricted dequeue
2. O/P restricted dequeue.

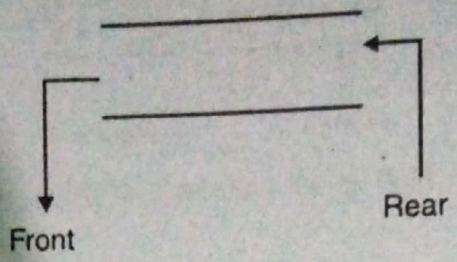
QINS (Q, F, R, ITEM)

1. If F = 1 and R = N, then :
   write : overflow and Return
2. If F = NULL then :
   Set F = 1 and R = 1
   Else if R = N, then :
   Set R = 1 [concept of circular array]
   Else
   Set R = R + 1

3. Set Q [R] = ITEM
4. Return.

Q 50. Define queue. How queues are implemented using double link list? (PTU, May 2006)

Ans. A queue is a linear list of elements in which deletions can take place only at one end called front and insertion can take place only at other end called rear end.



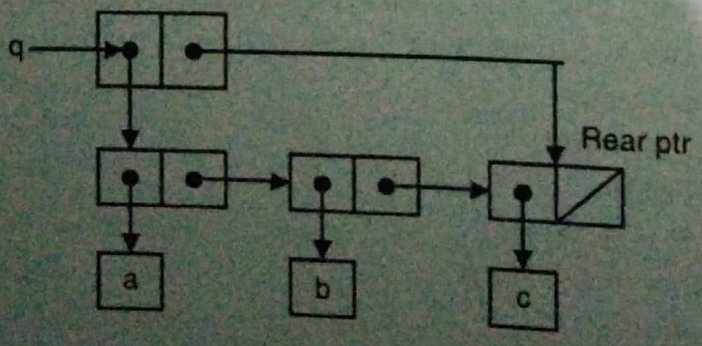
Queues implemented using Double link list are called dequeues. A dequeue is a linear list in which elements can be added or removed at either end but not in middle. Dequeue is maintained in memory by a circular array with pointers LEFT and RIGHT, which point to two ends of dequeue. Here DEQUE [1] comes after DEQUE [N]. There are two variations of a dequeue.

1. I/P restricted dequeue : It is a dequeue which allows insertions at only one end of the list but allows deletions at both ends of list.

2. O/P restricted dequeue : It is a dequeue which allows deletions at only one end but allows insertions at both ends of the list.

Q 51. What are the front and rear pointers of queue? (PTU, May 2013, 2011)

Ans. A queue is a sequence in which items are inserted at one end (called the rear of the queue) and deleted from the other end (the front). Fig. shows an initially empty queue in which the items a and b are inserted. Then a is removed c and d are inserted and b is removed.Because items are always removed in the order in which they are inserted, a queue is sometimes called a FIFO (first in, first out) buffer.



Operation	Resulting Queue
(define q (move-queue)	
(insert – queue ! q'a) a	
(insert – queue ! q' b) ab	

Queue operations

(delete – queue ! q) b
(insert – queue ! q' c) bc
(insert – queue ! q' d) bcd
(delete – queue ! q) cd

Q 52. Define queue. What are its various types?

Ans. A queue is an ordered collection of items from which items may be deleted at one end (called the front of the queue) and into which items may be inserted at the other end (called the rear of the queue).

Types of Queue : There are four types of queue :

1. Normal Queue/Linear Queue
2. Circular Queue
3. De-Queue
4. Priority Queue.

Q 53. Explain the various features of queue.

Ans. 1. A list structure will two access points called the front and rear.
2. All insertions occur at the rear and deletions occurs at the front.
3. Varying length.
4. Homogeneous components.
5. It has a (FIFO) first. In, first-out characteristic.

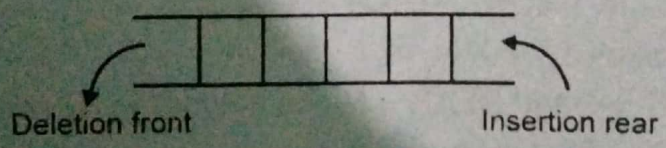
Q 54. How queues can be represented?

Ans. The queues can be represented or stored in computer memory by two ways :

1. By using arrays
2. By using linked list.

Q 55. Explain linear queue.

Ans. In linear queue no circular operation is performed that is last position is full and first cell is empty even then first position will not be occupied by an element. In linear queue, the overflow condition occurs when FRONT = 1 and REAR = N.



Q 56. What are the various operations performed on circular queue?

Ans. The following operations are performed an circular queue :

1. Create operation
2. Enqueue operation/insertion
3. Dequeue operation/deletion.

Q 57. Write insertion and deletion algorithms an linear queue.

Ans. Insertion/Enqueue Algorithm :

Enqueue (queue, item)

1. Check whether the queue is full or not. If the queue is empty then flash an error message
   "Queue Overflow" and
   goto step 5, otherwise go to step 2.
2. Increment the value of 'rear' as
   rear = rear + 1
3. Store the 'item' at rear postion as queue [rear] = item
4. Set front = 0 only when (front = = −1)
5. Exit.

Deletion/Dequeue Algorithm :

Dequeue (Queue)

Here 'queue' contains the base address of the array queue.

1. Check whether the queue is empty or not. If the queue is empty then flash an error message "Queue underflow" and go to step 4, otherwise go to step 2.
2. Access the front element as item = queue [front]
3. Reset the value of front and rear if (front = rear) ; otherwise increment the value of front = front + 1
4. Exit.

Q 58. What are queues and their applications? Write an algorithm to demonstrate insertion of a node in a queue. **(PTU, May 2019 ; Dec. 2011)**

Ans. Queue : A queue is a linear data structure in which new elements are inserted at one end and elements are deleted from the other end.

Applications :

1. The main application of queues is that it is used in traversing the graph using Breadth First Search (BFS) i.e. to find shortest path between two vertices.

2. Priority queues are used to assign priority s.t., an element of higher priority is processed before any element of lower priority.

An algorithm for insertion of an element in a queue.

Q INSERT(QUEUE, N, FRONT, REAR, ITEM)

Step 1: [Queue alread filled?]
If FRONT = 1 and REAR = N or if FRONT = REAR + 1, then :
Write : OVERFLOW and Return

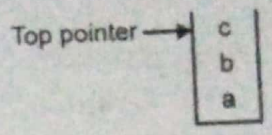
Step 2 : [Find new value of REAR]
If FRONT = NULL then [queue initially empty]
Set FRONT = 1 and REAR = 1
Else if REAR = N then
Set REAR = 1
Else
Set REAR = REAR +1
[End of If data structure]

Step 3 : Set QUEUE [REAR] = ITEM This insert new line

Step 4 : Return.

Q 59. What is a top pointer of stack? **(PTU, May 2013)**

Ans. Top pointer is a pointer to the stack from where operations are performed i.e. starting pointer where push and pop operations can be performed easily. .

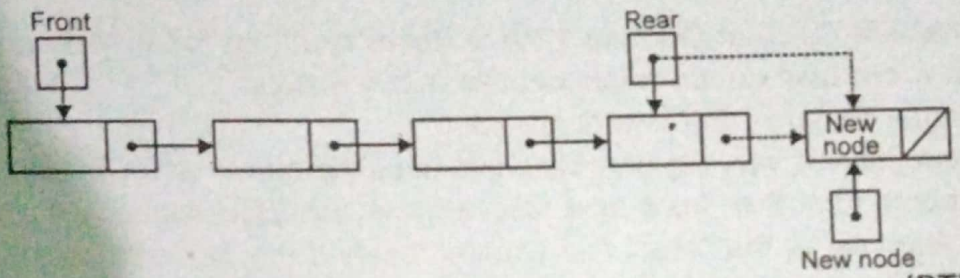


Q 60. Post-fix Expression. **(PTU, May 2014)**

Ans. In postfix expression if operator symbol is placed after its two operands, it is called postfix or reverse polish expression and this way of writing an expression is called as postfix or reverse polish notation of an expression.

Q 61. Linked representation of queue. **(PTU, May 2014)**

Ans.



Q 62. What is a circular queue and its use ? **(PTU, Dec. 2014)**

Ans. It is better than a normal queue because in this we can effectively utilize the memory space. If we have a normal queue and have deleted some elements from there then empty space is created there and even if the queue has empty cells than also we cannot insert any new element because the insertion has to be done from one side only and deletion has to be done from another side.

Application : Circular queue is used in VB.net
Polynomials
Sequential atomic operations

Q 63. Evaluate : (a) 1 24 3 + * 41 – (b) 25 7 * 14 – 6 + **(PTU, Dec. 2014)**

Ans. (a) 1 24 3 + * 41 –

Token read	Content of stack
1	1
24	1, 24
3	1, 24, 3
+	1, 27
*	27
41	27, 41
–	14
)	14

(b) 25 7 * 14 – 6 +

Token read	Content of stack
25	25
7	25 7
*	175
14	175, 14
–	161
6	161, 6
+	167
)	167

Q 64. What is the usage of stack in recursive algorithm implementation ? **(PTU, Dec. 2014)**

Ans. In recursive algorithms, stack data structures is used to store the return address

when a recursive call is encountered and also to store the values of all parameters essential to the current state of the procedure.

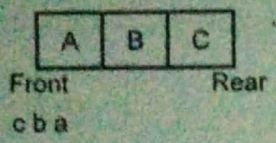
Q 65. What is the postfix form of the following prefix expression ? (PTU, May 2015)

***+ab–cd.**

Ans. ab+cd–*

Q 66. Assume that a queue is available for pushing and popping elements. Given an input sequence a, b, c, (c be the first element), give the output sequence of elements if the rightmost element given above is the first to be popped from the queue. (PTU, May 2015)

Ans.



c b a

Q 67. What are Circular Queues ? Write down routines for inserting and deleting elements from a circular queue implemented using arrays. (PTU, May 2015)

Ans. Circular Queue : *Refer to Q.No. 43*

Static queue have a very big drawback that once the queue is FULL, even though we delete few elements form the "front" and relieve some occupied space, we are not able to add anymore elements as the "rear" has already reached the Queue's rear most position. The solution lies in a queue in which the moment "rear" reaches its end; the "first" element will become the queue's new "rear". This type of queue is known as circular queue having a structure like this in which, once the Queue is full the "first" element of the Queue becomes the "Rear" most element, if and only if the "Front" has moved forward;

insert(queue, n, front, rear, item)

This procedure inserts an element item into a queue.

1. If front = 1 and rear = n, or if front = rear + 1, then :
   write : overflow, and return
2. [find new value of rear]
   If front = NULL, then : [queue initially empty.]
   set front = 1 and rear = 1.
   Else if rear = n, then:
   Set rear = 1.
   Else :
   Set rear = rear + 1.
   [end of structure.]
3. Set queue[rear] = item. [this inserts new element.]
4. Return.

delete(queue, n, front, rear, item)

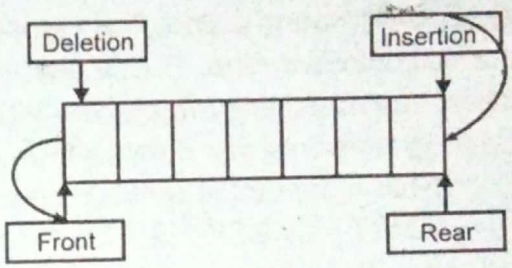
This procedure deletes an element from a queue and assigns it to the variable item.

1. [queue already empty?]
   If front = NULL, then :
   write : underflow, and return.
2. Set item = queue[front].

3. [find new value of front].
   If front = rear, then : [queue has only one element to start].
   Set front = NULL and rear = NULL.
   Else if front = n, then :
   Set front = 1.
   Else :
   Set front = front + 1.
   [end of structure.].
4. Return.

Q 68. Explain the terms Front and Rear for queue. **(PTU, May 2015)**

Ans. A queue is a non-primitive, linear data structure, and a sub-class of list data structure. It is an ordered, homogeneous collection of elements in which elements are appended at one end called **rear** end and elements are deleted at other end called **front** end. The meaning of front is face side and rear means back side.



Queue

Q 69. Explain application of Stack in recursive functions with example. **(PTU, May 2015)**

Ans. Recursive function uses stack. A stack is a Last In First Out (LIFO) data structure This means that the last data item to get stored on the stack (often called push operation) is the first data item to get out of the stack (often called pop operation). This is similar to that of stacking plates-the last plate that goes on the stack is the first one to get out of it.

- When function call itself, a new set of variables and parameters are stored on the stack and the function code is executed from the top with these new variables.
- As each recursive call returns, the old variables and parameters declared in the function are removed from the stack, and execution resumes immediately after the recursive call inside the function.
- The recursive function must have a conditional statement, such as if, somewhere to force the function to return without the recursive call being executed.

We can also say that, in the recursive function the call to self function is made. The recursive call means invoking the push operation, and when the control is returned from recursive function to main the pop operation is supposed to be performed. The recursion uses the concept of internal stack.

Q 70. What is the role of Priority queue in operating system design ? **(PTU, Dec. 2015)**

Ans. The typical example of priority queue is scheduling the jobs in operating system. Priority queues are used in well-known computer science application is job scheduling algorithms in the operating system design where the job or tasks with highest priorities have to be processed first. Typically operating system allocates priority to jobs. The jobs are placed in the queue and position of the job in priority queue determines their priority. There are three kinds of jobs.

1. Real time
2. Foreground
3. Background

The operating system always schedules the real time job first. If there is no real time job pending then it schedules foreground jobs. Lastly it schedules the background jobs.

Q 71. Why do we need Queues ? Write the algorithms/programs for EmptyQ, FullQ, InsertQ and DeleteQ operations. **(PTU, Dec. 2015)**

Ans. A queue is logically a first in first out (FIFO) type of list. In our everyday life we come across many situations where we ought to wait for the desired service, there we have to get into a waiting line for our turn to get serviced. This waiting queue can be thought of a queue. Queue means a line. For example : At the railway reservation booth, we have to get into the reservation queue. Thus a queue is a non-primitive data structure. It is an homogeneous collection of elements in which new elements are added at one end called the rear end and the existing elements are deleted from other end called the front end :

1. This algorithm is used to check whether a queue is empty or not
   EMPTY-CHECK (QUEUE, FRONT, REAR, EMPTY)
   1. If (FRONT=REAR + 1) then
      a . EMPTY : = true;
   2. Otherwise
      a . EMPTY : = false;
   3. Return
2. This algorithm is used to check whether a.queue is full or not.
   FULL-CHECK (QUEUE, FRONT, REAR, MAX, FULL)
   1. If (REAR = MAX) then
      a. FULL : = true;
   2. Otherwise
      a. FULL : = false;
   3. Return;
3. This algorithm is used to add or insert item to Queue.
   INSERT - ITEH (QUEUE, FRONT, REAR, MAX, ITEM)
   1. If (REAR = MAX) then
      a. Display "Queue overflow";
      b. Return;
   2. Otherwise
      a. REAR : = REAR + 1;
      b. QUEUE (REAR) : = ITEM;
   3. Return;
4. This algorithm is used to delete an item from queue.
   REMOVE - ITEM (QUEUE, FRONT, REAR, ITEM)
   1. If (FRONT = REAR + 1) then
      a. Display "Queue underflow";
      b. Return;

2. Otherwise
   a. ITEM : = QUEUE (FRONT);
   b. FRONT : = FRONT + 1;
3. Return;

Q 72. If the characters 'D', 'C', 'B', 'A' are placed in a queue (in that order), and then removed one at a time, in what order will they be removed ? (PTU, May 2016)

Ans. DCBA

Because As Queue follows FIFO structure.

Q 73. Suppose we have a circular array implementation of the queue with ten items in the queue stored at data [2] through data [11]. The current capacity is 42. Where does the insert method place the new entry in the array ? (PTU, May 2016)

Ans. data [12]

Q 74. What are the different representations of stacks ? How recursion functions can be implemented using stacks ? (PTU, Dec. 2016)

Ans. There are two ways to represent stacks in memory :

1. Array representation (Static data structure)
2. Linked list representation (Dynamic data structure)

A function calling itself or a call to another function, which in turn calls the first function is called a recursive function. These recursive function are executed using stacks. Return address and all local variables and formal parameters of the called method will be stored into the stack. Whenever any method is called all the elements stored in the stack will be restored after a return is executed. Recursion is implemented using the data structure stack.

Q 75. Explain how stack is applied for evaluating an arithmetic expression. (PTU, May 2017)

Ans. Evaluating Arithmetic expressions :

1. In order to evaluate an expression, standard precedence rules for arithmetic expression are applied.
2. The parenthesis are evaluate first followed by unary operator.
3. When unparenthesized operator of the same parenthesis are evaluated, the order of evaluation is from left to right except in the case of NOT (!), in which case the order is from right to left.

Q 76. Write an algorithm to convert infix to postfix expression and apply the same to convert the following expression from infix to postfix.
(a) (a * b) + c/d (b) (((a/b) – c) + (d * e)) – (a * c) (PTU, May 2018)

Ans. (a) (a * b) + c/d
(ab*) + c/d
ab * c/d +

(b) (((a/b – c) + (d*e)) – (a * c)
((ab/ – c) + (de *)) – (ac*)
ab/c – de * + ac * –

Q 77. Write the drawbacks of DQUEUES. (PTU, May 2018)

Ans. With the DQUEUES a complication may arise :

(a) When there is a overflow, that is, when an element is to be inserted into a dequeue which is already full or

(b) When there is underflow that is when an element is to be deleted from a dequeue which is empty.

❑❑❑