1. Sort the given array using Quick sort.
A[] : {5, 3, 2, 6, 4, 1, 3, 7}

Solution

Pass 1:

5	3	2	6	4	1	3	7
L,P							R

5	3	2	6	4	1	3	7
P			L			R	

5	3	2	3	4	1	6	7
P			L			R	

5	3	2	3	4	1	6	7
P					R	L	

1	3	2	3	4	5	6	7
P					R	L	

1	3	2	3	4	5	6	7

Pass 2:

1	3	2	3	4	5	6	7
P, L				R		P, L	R

1	3	2	3	4	5	6	7	
P	R	L				P, R,		L

Pass 3 :

1 | 3 | 2 | 3 | 4 | 5 6 7

L,P R

1 | 3 | 2 | 3 | 4 | 5 6 7

P R L

Pass 4 :

1 | 3 | 2 | 3 | 4 | 5 6 7

L,P R

Pass 5 :

1 | 3 | 2 | 3 4 5 6 7

P L, R

1 2 3 3 4 5 6 7

∴ 1 2 3 4 5 6 7.

2. Write the algorithm of Binary Search

Algorithm :

1. Start
2. Read the search element from the user
3. Find the middle element in sorted list.
4. Compare the search element with middle element.
   a. If both are matching, then display "Given element found" and terminate the function.
   b. If both are not matching, then check whether the search element is smaller or larger than the middle element.
   c. If Search element is smaller than middle element then Go To step-3 . for the left sub list of the middle element
      "Else"
      Go To step-3 for the right sub list of the middle element.
5. Repeat the same process until we find the search element in the list or until sub-list contains only one element.
6. If last element also doesn't match with the search element, the display "Element not found" and terminate the function.
7. Stop

3. Write the pseudo code of Merge sort.

Pseudo Code:

```
function mergesort (arr):
    if length (arr) <= 1:
        return arr
    mid = length (arr) / 2
    left = mergesort (arr [0 : mid])
    right = mergesort (arr [mid : length(arr)])
    return merge (left, right)

function merge (left, right)
    result = []
    while left and right are not empty:
        if left [0] <= right [0]
            append left [0] to result
            remove left [0]
        else
            append right [0] to result
            remove right [0]
    append remaining elements of left and right
        to result.
    return result.
```

4. Write down the pseudo code of implemendation of linked list as stack. (push & pop operation).

Push operation:

```
void push (item)
{
   NodeType *nnode ;
   int data ;
  nnode = (NodeType *) malloc (size of (NodeType));
   if (top==0)
  {
     nnode -> info = item ;
     nnode -> next = NULL ;
      top = nnode ;
  }
else {
     nnode -> info = item;
     nnode -> next = top ;
       top = nnode ;
}
```

Pop function!

```
        void pop ()
    {
        NodeType * temp ;
         if (top == 0 )
         {
            printf (" stack contain no elements : \n"
              return ;
         }
    else
      {
         temp = top ;
         top = top -> next ;
         printf ("\n deleted item is %d \t",
                     temp -> info ) ;
          free (temp) ;
      }
}
```

5. find the coefficient of x^4y^3 and the middle term in the expansion of $(2x+3y)^7$

Soln General term in the expansion of $(a+b)^n$, is given by.

$$T_{r+1} = \binom{n}{r} a^{n-r} . b^r$$

Here,

$n = 7$

$a = 2x$

$b = 3y$

r is the term number index.

So,

The general term becomes:

$$T_{r+1} = \binom{7}{r} . (2x)^{7-r} . (3y)^r$$

Simplify:

$$T_{r+1} = \binom{7}{r} . 2^{7-r} . x^{7-r} . 3^r . y^r$$

Coefficient of x^4y^3

On comparing:

$7 - r = 4$

$r = 3$

∴ Coefficient $= \binom{7}{3} 2^4 \cdot 3^3 = 15{,}120$

Middle term!