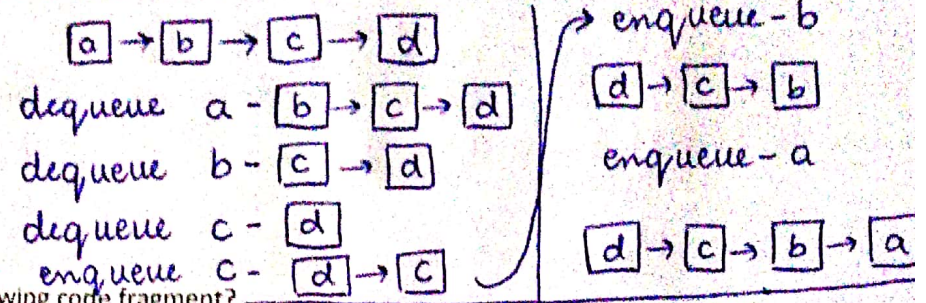
Solution

Data Structures and Algorithms
Quiz 2

Time Allowed: 20 minutes

Q 1) Mark the best choice: [5]

1) The initial configuration of the queue is a, b, c, d (a is the front end). To get the configuration d, c, b, a one needs a minimum of?
   a) 2 deletions and 3 additions
   b) 3 additions and 2 deletions
   (c) 3 deletions and 3 additions
   d) 3 deletions and 4 additions



2) What is the running time of the following code fragment?

```
10 - for(int i=0; i<10; i++)
N  -   for(int j=0; j<N; j++)
4  -     for(int k=N-2; k<N+2; k++)
          cout << i << " " << j << endl;
```

fixed, not depending on N

N+1 times check, N times inner loop

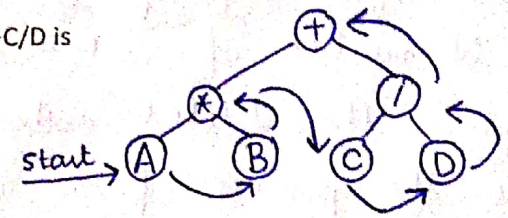
fixed, always Θ4 iterations.

a) O(log N)
b) O(N log N)
(c) O(N)
d) $O(N^2)$

$O(10*N*4) \Rightarrow O(40*N) \Rightarrow 40\ O(N)$

3) Which of the following is not the required condition for binary search algorithm?
   a) The list must be sorted
   b) There should be the direct access to the middle element in any sublist
   (c) There must be mechanism to delete and/or insert elements in list
   d) none of above

4) The postfix form of A*B+C/D is
   a) *AB/CD+
   (b) AB*CD/+
   c) A*BC+/D
   d) ABCD+/*



5) Consider following matrix:

1	0	9	8	3
4	5	6	7	8
2	3	4	5	6

Column major representation of this matrix will be:
   a) 1 0 9 8 3 4 5 6 7 8 2 3 4 5 6
   b) 1 4 0 5 2 3 9 8 6 7 4 5 3 8 6
   (c) 1 4 2 0 5 3 9 6 4 8 7 5 3 8 6
   d) 5 3 8 6 6 4 8 7 1 4 2 0 5 3 9

Q 2) Carefully observe the code below. Write the purpose of code in few lines and also show output and/or modified contents of the list. [10]

Assume that the single link list contains: 23->34->3->9->10->4->3->23. Following function deleteMN is called for M=3, N=2.

```
void deleteMN(struct node  *head, int M, int N)  {
        struct node *curr = head, *t;
        int count;
    while (curr)  {
        for (count = 1; count<M && curr!= NULL; count++)
                curr = curr->next;
        if (curr == NULL)
                return;
        t = curr->next;
        for (count = 1; count<=N && t!= NULL; count++)  {
                struct node *temp = t;
                t = t->next;
                delete temp;
        }
        curr->next = t;
        curr = t;
    }           // end while
}               // end function
```

(5)

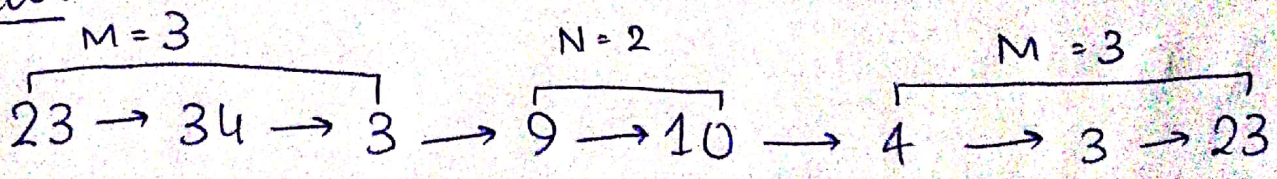
Purpose:

→ The function starts from the head of the linked list and move forwards. After skipping M number of nodes, it begins to delete N number of nodes. It then links the node that comes after the deleted nodes to the current node and continues the process from new position.

→ In short, it deletes N number of nodes after skipping M number of nodes.

(5)

Output:



Answer : 23 → 34 → 3 → 4 → 3 → 23

Q 3) An array is allocated dynamically for 10 integers to store a full binary tree: **[10]**

a) How many levels of the full binary tree can be stored in array A?

b) What should be the size of array to store a full binary tree with 32 leaves?

c) Write a C/C++ function to insert an element in a full binary tree. Function will take 4 parameters: element value to be inserted, an integer pointer, size of the array and number of total elements stored so far in the array. If feasible, insert the next element, otherwise re-size the array and then store the next element.

```
bool insert(int x, int *arr, int size, int count);
```

a) number of nodes $= 2^L - 1$ $\quad L \rightarrow$ number of levels

for $L=3$ $\quad$ for $L=4$

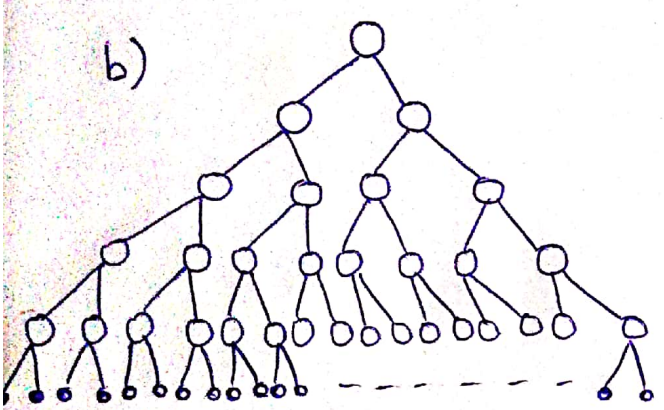
$2^3 - 1 \rightarrow 7$ $\quad$ $2^4 - 1 \rightarrow 15$

answer:

Since 15 exceeds 10, we can store only upto 3 levels.

---

b)



$\longrightarrow$ 32 leaves

number of leaves $= 2^{L-1}$

$32 = 2^{L-1}$

$2^5 = 2^{L-1}$

So $L = 6$

Total nodes $= 2^L - 1$

$= 2^6 - 1$

total nodes $= 63$

Size of array $= 63$ $\quad$ or simply $\left(\sum_{i=0}^{i=5} 2i \right)$

```
bool insert (int x, int * arr, int size, int count)
{   if (count < size)
    {   arr [count ] = x;
        count ++;
        return true;
    }
    int newSize = size * 2;
    int * newArray = new int [newSize];
```

```
for (int i=0 ; i < size ; i++)
        newArray[i] = arr[i];
delete [] arr;
arr = newArray;
size = newSize;
arr [count++] = x;
return true;
}
```