POSSESION OF MOBILES IN EXAMS IS UFM PRACTICE.

Name: _____ Enrollment No. 2210337

Jaypee Institute of Information Technology, Noida

T1 Examination, Even 2024

B. Tech. IV Semester / VI Semester (Minor Specialization)

Course Name: Algorithms and Problem Solving **Maximum Time:** 1 Hr.

Course Name: 15B11CI411 **Maximum Marks:** 20

CO1	Demonstrate a familiarity of complexity classes, the notion of algorithm, asymptotic analysis, and problem solving approaches.
CO2	Apply a standard algorithm for solving fundamental problems such as sorting, searching, and graph based problems.
CO3	Analyze and identify an appropriate data structure and/or algorithm design strategy for a given problem.
CO4	Design an efficient algorithm to solve a given problem.

Q1. In terms of Big-Oh notation, find the time complexity of the algorithm **F1** (Fig.1). **[CO1 (Understand); 3 Marks]**

```
Algorithm F1(int N) {
   int B = 0;
   for(int I = 0; I < N; I++) {
      int A = 0;
      for(int J = N; J > 1; J = J / 2) { A = A + 1; }
      for(int K = 1; K < A; K = K * 2) { B = B + 1; }
   }
}
```

Fig.1: Given algorithm, F1

Q2. Ternary search is a searching algorithm that is used to find the position of a target value within a sorted array. Here, we compare the target value with elements at two points that divide the array into three equal parts. Propose the algorithm for ternary search to find the position of a target value within a sorted array. **[CO2 (Apply); 4 Marks]**

Q3. You have been given the pieces of papers along with their length (L) and width. Width of all the papers are same, however length of all the papers are unique. These papers are identified by unique numbers (P) as shown in Fig. 2 (Paper Number: 1, 2, 3, 4, and 5 and Length as 1.0, 1.5, 2.8, 2.0, & 2.5):

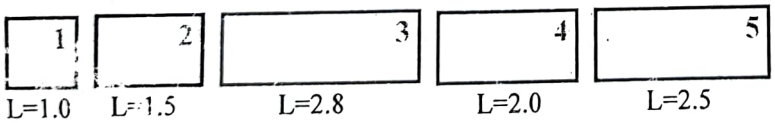

Fig. 2: An example having 5 papers, numbered as 1, 2, 3, 4, and 5.

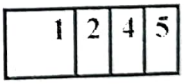

Fig. 3: Possible Solution

You need to arrange these papers by keeping one paper (I^{th} paper) on another paper (J^{th} paper) such that: $L_I < L_J$ and $P_I < P_J$. In other sense, P_J completely overlaps P_I. Surely, you cannot arrange all N papers fulfilling the mentioned conditions. However, fulfilling the mentioned conditions, it is desired to find out the maximum number of papers which can be arranged. For the example given in Fig. 1, maximum, 4 papers can be arranged (Fig. 3): Paper No. 1, 2, 4, 5 (*i.e.* $P_I < P_J$) having Length as 1.0, 1.5, 2.0, 2.5 (*i.e.* $L_I < L_J$). In other sense, Paper 5 overlaps Paper 4; Paper 4 overlaps Paper 2; and Paper 2 overlaps Paper 1.

(a) Propose a backtracking based **non-recursive algorithm** to find out the maximum number of papers which can be arranged as per the given conditions. Input to your algorithm is list of papers with their unique sequence numbers and unique lengths. Draw the state space tree for above example of 5 papers (Paper Numbers: 1, 2, 3, 4, & 5 and Length of papers as 1.0, 1.5, 2.8, 2.0, & 2.5 respectively). **[CO2 (Apply); 6 Marks]**

(b) Mention the data structures used in the proposed **algorithm.** **[CO3 (Analyze); 1 Marks]**

Q4. An algorithm/function, F2 is given in Fig.4. **[CO2 (Apply); 3+2+1=6 Marks]**

```
int * F2(int *A, int N) // A is an integer array and N is the count of elements in A
{
  int *B, T, I, J, M;
  if (N == 2 && A[0] > A[1]) { T = A[0]; A[0] = A[1]; A[1] = T; return A; }
  else if(N > 2)
{
     M = ceil (2 * N / 3.0); // ceil function returns the ceiling value, e.g. ceil(5.4) returns 6
     B = new int[M];
     J = 0;
for(I = 0; I < M; I++) { B[J++] = A[I]; }
B = F2 (B, J);
     J = 0;
for(I = 0; I < M; I++) { A[I] = B[J++]; }
     J = 0;
for(I = N – M; I < N; I++) { B[J++] = A[I]; }
B = F2 (B, J);
     J = 0;
for(I = N – M; I < N; I++) { A[I] = B[J++]; }
     J = 0;
for(I = 0; I < M; I++) { B[J++] = A[I]; }
B = F2 (B, J);
     J = 0;
for(I = 0; I < M; I++) { A[I] = B[J++]; }
     return A;
}
  return A;
}
```

Fig.4: Given algorithm/function F2

In context of the given algorithm/function, F2, answer following:

(a) Which problem is solved by the algorithm F2. Write the name of the problem.

(b) Let us consider the contents of the array, ARR[] = {8, 7, 6, 4, 3}. It is passed into F2 as F2 (ARR, 5), where 5 is the count of elements in the array, ARR. After the execution of the function, F2, what will be the updated (if any) contents of the array, ARR. Considering the initial calling of the function, draw the recursion tree/solution tree depicting all the sub-problems and corresponding array elements.

(c) Write the recurrence equation/relation for the algorithm F2.