THE UNIVERSITY OF THE SOUTH PACIFIC

CS214: Design & Analysis of Algorithms
SCIMS, FSTE

Final Examination
Semester II, 2020

F2F/Blended Mode

Duration of Exam: 3 hours + 10 minutes

Reading Time: 10 minutes

Writing Time: 3 hours

Total Marks: 40

The exam covers the following learning outcomes:

CLO – 1: Evaluate the efficiency of algorithms (32.5%)

CLO – 2: Assess the suitability of different algorithms/data structures for solving a given problem (17.5%)

CLO – 3: Solve computationally difficult real world problems using appropriate algorithmic techniques (50%)

Instructions:

1. Write your answers in the space provided in this Question Paper.
2. Answer all questions. There are 10 questions and all questions are compulsory.
3. This exam is worth 40% of your overall mark. The minimum mark to pass the final exam is 16/40.
4. The total number of pages including this cover sheet is 15.
5. This is a closed book exam. No printed materials and electronic devices are allowed.

ID:_____	**Name:**_____
SeatNo: _____	**Campus:** _____

This page is intentionally left blank

1. Why is it important to use an appropriate data structure in a given algorithm? Explain by giving example(s). Is there any one-size-fits-all kind of data structure? (1+1+1 = 3 marks)

2. Determine the worst case time complexity of the Travelling Salesman Problem using the brute force approach? Show your working. (2 marks)

3. Determine the best, worst and average time complexity of following code: (2 marks)

```
int a = 0;
for (i = 0; i < N; i++) {
    for (j = N; j > i; j--) {
        a = a + i + j;
    }
}
```

4. Explain how empirical testing is done to determine the efficiency of a given algorithm or to compare its results with other algorithms of the same class? (3 marks)

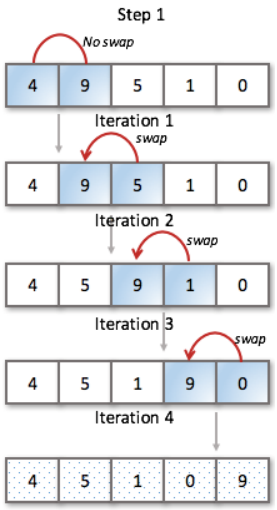
ID: _____

5. A variation of the *bubble sort* algorithm and an example illustrating what it does are shown below. Using this algorithm answer the following questions:

```
static void bubbleSortSimple (int S[]){

    int n = S.length;
    int temp;

    for (int j = 1; j < n; j++) {
        for ( int i=0; i < n-1; i++ ) {
            if (S[i+1] < S[i]) {
                temp = S[ i ];
                S [ i ] = S [ i+1 ];
                S [ i+1 ] = temp;
            }
        }
    }
}
```



• • •

5.1) Evaluate the Big O order of the best and worst time complexity for this algorithm? Show all working. (2 marks)

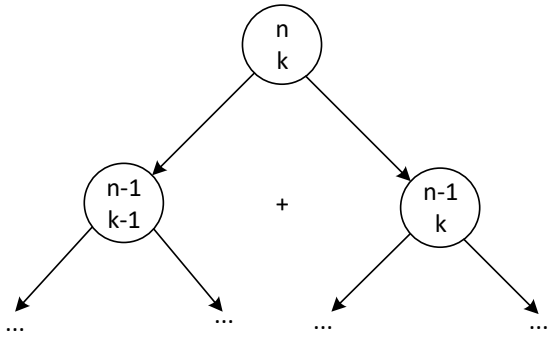
5.2) Under what condition(s) you will attain the best and worst time complexities? (2 marks)

(answer both questions 5.1 and 5.2 here)

6. Determine is the worst (or average case time complexity with +1 bonus marks) for the calculation of Binomial Coefficient using greedy approach? Are you satisfied with the order of your solution? If not, then how would you improve it further? Justify. (4 marks)

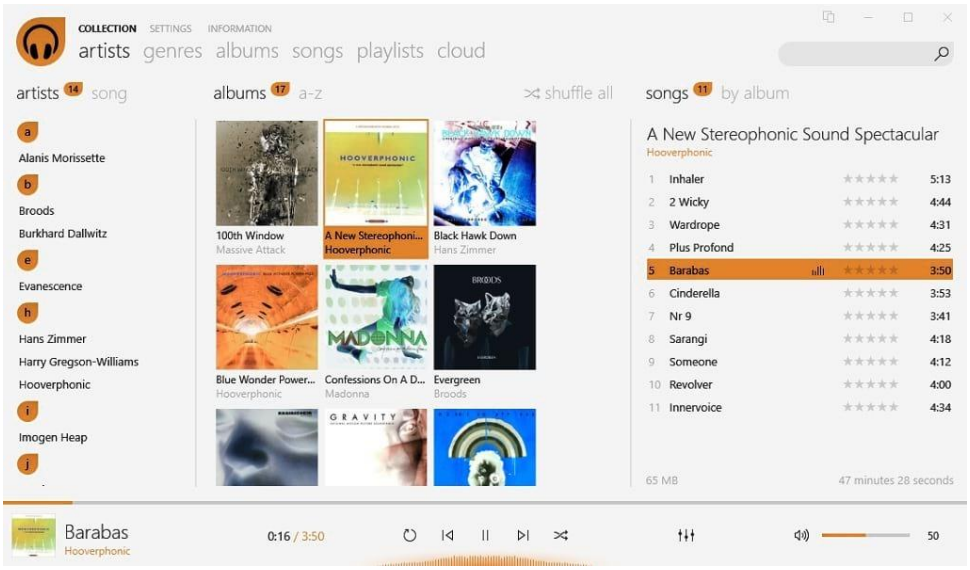
```
int bin (int n, int k){
    if (k == 0 || n==k)
        return 1;
    else
        return bin(n-1, k-1) + bin(n-1, k);
}
```

Hint: the worst case would be when k=n/2. You may start your analysis by completing the breakdown of the recursive equation as given below and identifying the pattern.



ID: _____

7. A music player is used to play one song at a time. Songs in the music player are linked to the previous and next song. you can play songs either from the starting or end of the list. A song can be either added or removed from the list. Choose the appropriate data structure that is most suited for this purpose and justify your choice? (2 marks)

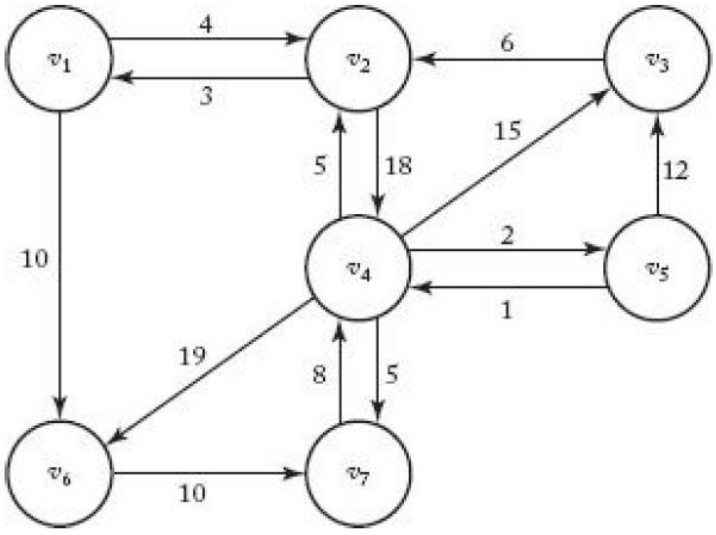


8. Floyd's algorithm given below in Figures 1 (a) has generated the path table in Figure 1 (c) from the given graph in Figure 1 (b) to determine the shortest path. Apply the Floyd's algorithm to determine the shortest path and shortest length from node v1 to node v5 and v1 to v3 using the principle of optimality? Show all workings. (4 marks)

```
void floyd2 (int n,
             const number W[][],
                   number D[][],
                   index  P[][])
{
  index i, j, k;

  for (i = 1; i <= n; i++)
     for (j = 1; j <= n; j++)
        P[i][j] = 0;
  D = W;
  for (k = 1; k <= n; k++)
     for (i = 1; i <= n; i++)
        for (j = 1; j <= n; j++)
           if (D[i][k] + D[k][j] < D[i][j]){
              P[i][j] = k;
              D[i][j] = D[i][k] + D[k][j];
           }
}
```

(a)



(b)

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

(c)

Figure 1: (a) Floyd algorithm, (b) a sample graph and (c) path table

9. Apply the Huffman's Algorithm to the encode (zip) a text file containing "to be or not to be" (exclude spaces for simplicity). Show all working. (5 marks)

10. Consider the 0-1 knapsack problem with four different item having different weight and profit value;
    item 1 (2kg, $3)
    item 2 (5kg, $5)
    item 3 (3kg, $5)
    item 4 (8kg, $9)
    item 5 (1kg, $3)

The knapsack has a capacity of 8 kg.

10.1) Apply greedy approach to compute the optimal profit in filling the knapsack (3 marks)

10.2) Apply a dynamic programming approach to solve the same problem. Show the complete working. (3 marks)

10.3) Apply the backtracking approach for the above knapsack problem. Show the State Space tree. (2 marks)

10.4) Develop Branch and bound solution for the given knapsack problem. Use the best first search approach. (3 marks)

END