**1.** Assume that you are a surf teacher and work in a surf school for a limited time: *m* days. You need to cover your expenses for days in which you have no lessons. You know in advance that if you are at this surf school on the *ith* day of your arrival, you will save *pi* TL where *pi* could be positive (if your earnings are higher than your expenses) or negative (otherwise). To maximize your earning, you have to choose carefully which day you arrive and which day you leave; the days you work should be consecutive and you don’t need to work for an entire month. For example, if m = 7 and p1 = −5 , p2 = 12 , p3 = 8 , p4 = -7 , p5 = 5 , p6 = 14 , p7 = 2 then if you worked from day 3 to day 6, you would earn (8 - 7 + 5 + 14) = 20 TL.

a) Develop an algorithm for this problem running in *O (m log m)* time. If your algorithm takes longer time, you will get a partial credit. Note that your algorithm should guarantee to find the exact optimum solution. Write a pseudo-code. (10 points)

b) Develop an *O(m)* time algorithm and write its pseudo-code. (10 points)

**2)** You will design a process scheduler for CPU. The scheduler will serve sequentially for *n* processes *P1,...,Pn* . The process *i* keeps busy CPU for a duration (dtime[i]). Design an algorithm that produces an optimal process schedule, in which each process receives a service start time (stime[i]) to be served for the necessary amount of time, and this algorithm minimizes the average wait of all processes:

Assume that the CPU and all processes are ready to start at time 0.

**Input:**

***n***: The number of processes.

***dtime [1...n]*** such that for all *i, 1 ≤ i ≤ n,* ***dtime [i]*** is the duration time of process ***i*** (in milli seconds).

**Output:**

***stime[1...n]*** such that for all *i, 1 ≤ i ≤ n,* ***stime[i]*** is the time (milli seconds) when process ***i*** will start to be served.

1. Explain in detail the algorithmic strategy to solve this problem. (5 points)
2. Write a pseudo-code of your algorithm. You should keep track of the processes, so that *stime[i]* corresponds to the same process *i* whose duration was *dtime[i].* Explain the total running time of your algorithm. (15 points)

**3)** Given a list of numbers *a1, . . . , am* an increasing subsequence is a list of indices *i1, . . . , ik ∈ {1,...,m}* such that *i1 < i2 < ... < ik* and *ai1 ≤ ai2 ≤ ,..., ≤ aik*. The longest increasing subsequence is the longest list of indices with this property.

For a given list of numbers: 2,1,5,3,4,7.

The longest increasing subsequence: 1,3,4,7 (note: there would be several subsequences)

a) Suppose that a greedy algorithm chooses the first number in the list, and then repeatedly chooses the next number which is larger. Is it a correct strategy? Either prove its correctness or provide a counter example. (2 points)

b) Suppose that a greedy algorithm that chooses the smallest number in the list, and then repeatedly chooses the smallest number which comes after this selected one. Is it a correct strategy? Either prove its correctness or provide a counter example. (2 points)

c) Develop a dynamic programming solution for this problem. Write a pseudo-code of your solution. Explain the total running time of your algorithm. (11 points)

**4)** Using the given list of sorting algorithms, choose the best (i.e., most efficient) for each scenario to reduce the expected running time. Explain your choice briefly. Answers without explanation will not get any points.

**Insertion Sort - Quick Sort - Heap Sort - Merge Sort**

1. The original data is stored in a cloud server, your sorting function runs in a local server. So, whenever you need reach some data, you will use the internet connection whose speed is quite slow. Hence swapping two elements in your sorting function has a high cost. Your aim is to minimize the cost of swap operations as much as possible. (5 points)
2. You will develop a website in which users can upload their data and request the ordered version of their data. The input could be small, large, almost ordered, or totally mixed. Based on such an unknown nature of the input data, you should provide a fast running website. (5 points)
3. You bought a new playing cards having 52 individual cards. You want to sort them based on their shapes and numbers. Some of the cards are not in their correct order in the deck. Which sorting algorithm should be used so that only one or two cards are taken from the deck at any time? (5 points)