FIT2004 S2/2016: Assessment questions for week 6

THIS PRAC IS ASSESSED! (6 Marks)

DEADLINE: Monday, 29-Aug-2016 10:00:00 AM
CLASS: This programming exercise has to be completed before the deadline. Your demonstrator will check your submission during the lab. He will ask you a series of questions to assess your understanding of this exercise, and gauge how you implemented it. It is required that you implement this exercise strictly using **Python programming language**. Practical work is marked on the performance of your program **and also on your understanding of the program**. A *perfect* program with zero understanding implies you will get **zero** marks! "Forgetting" is not an acceptable explanation for lack of understanding. Demonstrators are not obliged to mark programs that do not run or that crash.
After/befor your demonstrators have interviewed you, you are expected to work towards the programming competition or other questions that will be provided to you on the day.
SUBMISSION REQUIREMENT: You will need to submit a zipped file containing your Python programs (named salesman.py and playlist.py) as well as a PDF file briefly describing your solutions and their space and time complexities. The PDF file must give an outline of your solution (e.g., a high level idea of how did you solve it) and the **worst-case** space and time complexity of your solution. The zipped file is to be submitted on Moodle before the deadline.
Important: You should carefully consider the border cases and make sure that you return correct results for all cases. Although you are not required to include the proof of correctness in your submissions, you are encouraged to formally prove the correctness of your programs. This will help you in identifying and fixing the bugs in your program if present.

1 Task 1: Salesman's Dilemma

A door-to-door salesman is visiting a street and found that people hate their neighbors. He visited every house and everyone in the town told him that they are willing to buy some items from him but only if he does not sell anything to any of his neighbors. He noted down the items the residents of each house are willing to buy. Specifically, for each house i, he has recorded the total price of the items, p_i, residents of the house i will buy if he does not sell anything to any of the neighboring houses of i. He told everyone that he will come back after thinking about it.

He came to you asking for help in maximizing his sale. Specifically, he wants you to write a Python program that determines a strategy to maximize his sale. The program must print the house numbers of all the houses he must sell to (in ascending order of house numbers). Your program must also output his sale if he follows your advice.

But you ask him, "How do I know the neighboring houses of a house". He tells you that there are N houses along the street and are sequentially numbered 1 to N. A house numbered

i is called the neighboring house of a house j if $|i-j| \leq k$ where k is a parameter set in your program and $|i-j|$ denotes the absolute difference between i and j. For example, if $k = 3$, the house number 50 is called the neighboring house of all the houses numbered 47 to 53 and the house number 2 is the neighboring house of all the houses numbered 1 to 5. If $k = 1$, the house number 50 is the neighboring house of 49 and 51, and the house number 2 is the neighboring house of 1 and 3.

Your goal is to write an efficient Python program to help him. This problem can be solved using $O(N)$ space and $O(N)$ time. Your program must be named salesman.py.

1.1 Input

The input file consists of 2 lines. The first line of the input is the value of N corresponding to the total number of houses in the street. The next line contains N space separated numbers where $i-th$ of the numbers is p_i denoting the total price of the items the residents of house i are willing to buy if he does not sell to any neighboring house of i. Below is a sample input.

```
10
50 10 12 65 40 95 100 12 20 30
```

The above input shows that there are 10 houses. The first house will buy items worth of 50 and so on.

Important: Your program must read data from the input file named "houses.txt" (provided in the zipped folder). You will lose marks if the program does not correctly read from the provided "houses.txt' file. You must test your program extensively on larger data sets because it is quite possible that an incorrect algorithm may get correct results on this very small sample file. Your program will be tested on a file containing a much larger data set.

1.2 Output

The output must consist of two lines. The first line must give the house numbers he must sell to in order to maximize his sale. The house numbers must be printed in ascending order. The second line must print the total sale if he sells to these houses. Below is a sample output for $k = 1$ for the above input file.

```
Houses: 1 4 6 8 10
Total sale: 252
```

Below is a sample output for $k = 2$ for the above input file.

```
Houses: 1 4 7 10
Total sale: 245
```

Below is a sample output for $k = 3$ for the above input file.

```
Houses: 1 6 10
Total sale: 175
```

Note that k is a variable set in your program and it can be any value up to N, i.e., $k \leq N$. If there are more than one possible answers (e.g., multiple ways to achieve the maximum sale), you can print any of the answer.

2 Task 2: Playlist

Alice drives to work and listens to her favorite songs on her way. She hates it when she is in middle of a song and reaches her destination. This is because if she does not listen to a song till the end then it keeps playing in her mind for the whole day which affects her performance. She does not want to sit in the car waiting for the song to finish. Also, she does not want to stop listening to the song before she has arrived her destination. In other words, she wants the songs to play such that a song ends exactly when she has arrived her destination.

She is writing a phone app to help scheduling the songs. The app communicates with Google Maps app and determines the estimated time T (in seconds) to reach the destination. We will assume that the estimate T is perfect and it will take her exactly T seconds to reach the destination. Her app also has obtained the duration (in seconds) of each song in her phone. However, she does not know how to proceed further. She knows you are taking a course on Algorithms and Data Structures and you had helped her with the Celebrity question a couple of weeks ago. So, she is hoping you will be able to help her again.

Your goal is to write an algorithm that determines a playlist of the songs such that, if the songs are continuously played during her journey, a song finishes exactly when she reaches the destination. In other words, if the estimated duration of her journey is T seconds, the total duration of the songs in the playlist must also be exactly T. Furthermore, she does not want any song to be repeated. Therefore, your playlist should not have duplicate songs.

You are given a list of songs numbered 1 to N (called ID of the song). A song with ID i has a duration d_i (in seconds). You can assume that no two songs have the same duration. The output must print the playlist meeting the above requirement (in ascending order of the IDs of the songs). If it is not possible to find such playlist, you must report a message stating "Bad luck Alice!".

Your program must run in $O(NT)$ time with $O(NT)$ space where N is the total number of songs and T is the estimated journey time. Your program must be named `playlist.py`.

2.1 Input

The input file consists of 2 lines. The first line of the input contains a single integer N that represents the total number of songs in the phone. The next line contains N space separated numbers where $i-th$ of the numbers is d_i denoting the duration of the song with ID i. Below is a sample input.

```
5
10 3 5 7 2
```

Important: Your program must read data from the input file named "songs.txt" (provided in the zipped folder). You will lose marks if the program does not correctly read from the provided "songs.txt' file. You must test your program extensively on larger data sets. Your program will be tested on a file containing a much larger data set.

2.2 Output

Your program must have a variable T. Your output must print the playlist that meets the above requirements, in ascending order of IDs, with the duration of each song displayed next

to the song ID (see below). If there are more than one correct answers (multiple playlists with the total duration T), you are free to print any of the playlists. If it is not possible to find such a playlist, you must display "Bad luck Alice!"

Below is a sample output for the above input file where T is 14.

```
ID: 3 Duration: 5
ID: 4 Duration: 7
ID: 5 Duration: 2
```

Below is a sample output for 17.

```
ID: 1 Duration: 10
ID: 4 Duration: 7
```

Note that the following answer for $T = 17$ is also correct.

```
ID: 1 Duration: 10
ID: 3 Duration: 5
ID: 5 Duration: 2
```

Below is a sample output for the above input file where T is 16.

```
Bad luck Alice!
```

```
-=o0o=-
  END
-=o0o=-
```