Lab 11 – CSE 101 (Spring 2020)

Objectives

The primary objectives of this lab assignment are:

- To get experience working with Huffman trees and the Huffman encoding
- To get practice converting between different encodings
- To get more Python programming practice

1. Huffman Tutorial Project (1 point)

Download lab11.py. Go through the steps T71 to T91 in Chapter 8 of the Conery Textbook, adding that code to the file.

After each step, be sure to print the results and try to understand what is happening.

2. Converting Strings to Hexadecimal (2 points)

In the lab11.py file, implement the `print_hex` function that will print the ASCII codes in a string using hexadecimal notation:

```
>>> print_hex("Hello")

48 65 6C 6C 6F
```

Write your own code to convert each character to hexadecimal – do not use the hex() function. To do this, you can follow the same algorithm we used for converting a decimal number to binary by dividing number by 2 and looking at the remainder, except you need to divide by 16 since it is going to a base 16 number, instead of base 2.

You can use the ord() function to get the decimal value of a character. For example, ord("H") returns 72.

You will also want to use the provided `decimal_to_hex_string_dict`, which will allow you to map a decimal number between 0-15 to the hexadecimal string value.

It may be helpful to review the Chapter 8 slides and check out the `data_rep.py` sample code for Chapter 8.

3. More coding practice (2 points)

When you are applying for programming jobs, sometimes companies conduct online tests to determine your programming skills as part of the candidate screening process. There are several platforms for testing programming skills online, such as Codility and HackerRank. You can get the feel of some of the problems at the following page:

https://app.codility.com/programmers/lessons/

During this lab session you will solve the following problem from the Codility platform:

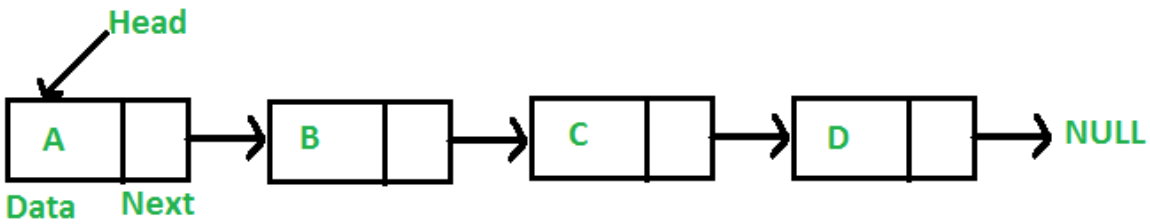
MaxProfit problem (Save your solution as `maxprofit.py`):
https://app.codility.com/programmers/lessons/9-maximum_slice_problem/max_profit/

If you enjoy solving this style of problems, you can try out other examples on the site as additional practice learning programming over the break.

4. Extra credit problem (2 points)

For this problem download

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:



In simple words, a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.

Refer to the class `Node` and class `Linked_list` in linked_list.py which simulate a node in the linked list and the linked list itself. Complete the implementation of `append_data` function in the provided linked_list.py file.

3. Submission

Submit the following programs on blackboard:

1) Completed `lab11.py` program
2) `maxprofit.py` program
3) `linked_list.py` program (optional, for extra credit)