Due: 2022/5/7

Homework 12 - Object-Oriented Programming

$*$ If there is any problem, please contact TA.

Name:____________ Student ID:____________ Email: ________________

Please include your source code file and a screenshot of execution result for each program problem in your submit.

Problem 1. Use Ada[1] language to implement a max heap (of int type) data type by array, it should have following methods:

- empty:whether it is an empty heap
- size:return the size of the heap
- push:add a element into heap
- pop:remove the top element from heap
- top:access to the top element in heap

For testing:
1.push 3
2.push 1
3.push 2
4.call size(), print size
5.call top(), print result
6.call pop()
7.repeat 5
8.call empty()
9.repeat 4

Problem 2. Suppose we have two strings: $x = CTACCG$ and $y = TACATG$. After an alignment, we have:

$$\begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ C & T & A & C & C & - & G \\ - & T & A & C & A & T & G \end{array}$$

Here ”−” means empty. In order to compare the distance between two strings, we usually use edit distance. There are two kind of penalties in edit distance.

[1]You can use this online website: **https://www.tutorialspoint.com/compile_ada_online.php**

- Gap: If in alignment, we match a character with empty(such as position 1 and 6), there will be a gap penalty.
- Mismatch: If in alignment, we match a character with a different character(such as position 5), there will be a mismatch penalty.

The alignment cost C is define as:

$$C = \sum_{gap\ pair} Gap\ Penalty + \sum_{mismatch\ pair} Mismatch\ Penalty$$

Suppose in this case, the gap penalty is a and the mismatch penalty is b, then the cost for this alignment is:

$$C = 2a + b$$

In **String Similarity** problem, we want to find the minimum alignment and its cost for two strings. We can use dynamic programming to solve this problem in $O(mn)$ space and $O(mn)$ time, where m and n are the length of the string. Actually there is a better algorithm called **Hirschberg Algorithm** which can solve this problem in $O(m+n)$ space and $O(mn)$ time.

The idea of Hirschberg Algorithm is we can combine divide and conquer with dynamic programming. Suppose we have two strings: $x_{1,\ldots,n}$ and $y_{1,\ldots,m}$. First we divide string x into two sub strings: $x_{left} = x_{1,\ldots,n/2}$ and $x_{right} = x_{n/2+1,\ldots,n}$. Then we divide string y at position k into two sub strings: $y_{left} = y_{1,\ldots,k}$ and $y_{right} = y_{k+1,\ldots,m}$. We calculate the costs for (x_{left}, y_{left}) and (x_{right}, y_{right}) by divide and conquer.

$$C_k = C(x_{left}, y_{left}) + C(x_{right}, y_{right})$$

We traverse k from 0 to m to find the optimal k_{opt} which leads to the minimum cost.

$$k_{opt} = \min_{k=0,\ldots,m} C_k$$

In this problem, you are to find the lowest alignment cost between 2 string sequences. There are only uppercase letters in strings. The mismatch penalty of two letters are the distance between their order in ASCII. For example, $Distance(A, E) = 4$. The gap penalty is fixed as 7.

a) Implement a Hirschberg Algorithm to solve this problem with Smalltalk[2].

b) Test your program with following inputs:

$$x = CTACPG$$
$$y = TACATG$$

Your program should print the lowest cost and the aligned strings(add ”−” into the strings to represent empty)

[2]You can use this online website: **https://www.tutorialspoint.com/execute_smalltalk_online.php**