**Programming Assignment 1**

**Name: Rui Zuo**

**x500: zuoxx067**

**ID: 5323641**

**Class Number: CSCI 4041**

**Professor Name: Nikolaos Papanikolopoulos**

1. Location and Name of Files

Source codes of three questions are respectively name as partA.py, partB.py and partC.py. And the locations of them and corresponding txt files is in a file folder named as Rui\_pa1

2）Priority Queue

2.1 Program Implementation Description

My program is defined in a class named banker and included all the corresponding functions. A read\_file function is called to read the input of target txt file. Lines of strings in the txt file will be separated and striped into name and priority number. Finally, all these data will be stored in the global class variable named queue. Then the function make\_heap will be called in read\_file to make a max heap towards queue. The Heapify function will be used in the process of making heap. Besides, the required function could be called by the instance of class banker. Function Maximum will return the list which contains the name of the customer and its priority number. Function Extract\_Max will also return the same result but the data of this customer will be removed from the queue and the fuction make\_heap will be called again. Function Insert could accept a list which contains the name and priority of new customer. This list will be added into the queue. Similarly, function make\_heap will be called again to maintain the heap.

2.2 Known Bugs

None to the best of my k

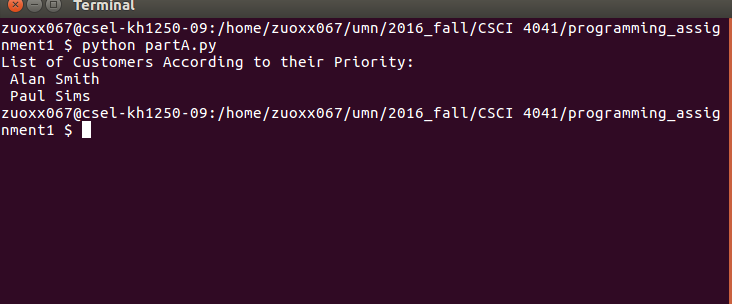
2.3 Style Guidelines

My program follows the general indentation style of python 2.7. Every function within the class have been followed with comments. My variables were named by its definition and usage separated by a underline.

2.4 Innovations

I stored my data of customers into a sublist which has two elements. One is the name and the other one is the priority number. For example, if I have 10 customers the list variable queue of class banker will contain 10 sublists which store all the data extracted from the file

2.5 Terminal Session and Source Code



3）Huffman

3.1 Program Implementation Description

My define two classes which are Huffman and HuffmanNode. HuffmanNode is the node of the Huffman tree which has 5 variables. Class Huffman contains the function read\_file which is called to process the input from input file. It will count the frequency of each appeared character and store them into a class global dic named stat. Then the global list data belons to the class will be used to store the information of stat within its sublist.. After data from input file is processed done. The function create\_tree will be called to make a Huffman tree. In the process of making a tree, function make\_node will be called first to create nodes for each characters. The weight of class Huffmannode is the frequency of the character. To sort the node, a list called nodelist will be used to store the node. The insertion sort will be implemented with the nodelist.

Every time we choosed frist two nodes in the nodelist and create a new node as the parent node of these two nodes The new node will be put into the nodelist and the precess will be repeated until the tree is successfully created.

3.2 Known Bugs

I have noticed that when I read the whole file into a list, the amount of lines will be

than the actual amount of lines. The program may be broken caused by this bug.

Besides, the frame of printed tree is a little confused because of the space between sibling nodes. I try my best to print out the tree whose structure is following the requirement strictly. But it failed.

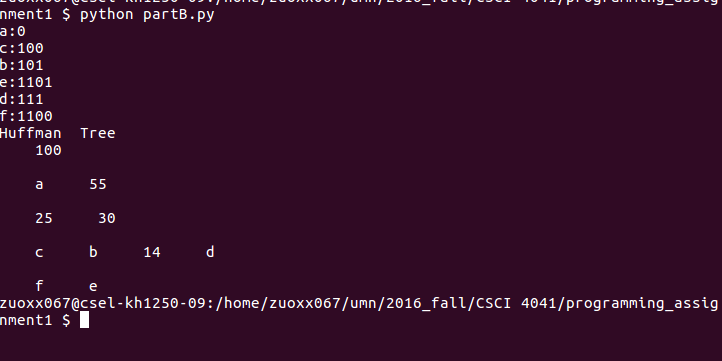
3.3 Style Guidelines

My program follows the general indentation style of python 2.7. Every function within the class have been followed with comments. My variables were named by its definition and usage separated by a underline.

3.4 Innovations

To successfully print out the frame of the Huffman tree. I design the function named depth\_deter. It is conduct a level\_order transversal to the Huffman tree. We know that that the level\_order transversal require a queue to store the nodes. Every time I put the child node into the queue, I set the depth of the nodes as the depth+1 compared with the depth of the parent node. So the depth of nodes in the tree is determined. After this function is determined. We could smoothly print out the Huffman tree . When we print the tree, every time we find that the depth of the node to be printed out is incremented, we move to next line. That is the logic of function print\_tree

3.5 Terminal Session Script and Source Code



4) KMP

4.1 Program Implementation Description

I design this program follow the general design of the kmp algorithm. A class named kmp which contains the function read\_file which is called to to read the input file and extract pattern string and matching string. Function make\_next is called to make the next table for the pattern string. It will be called in the function kmp\_matching. Finally the start position will be printed as the requirement.

4.2 Known Bugs

For the convenience, the string list in my program which contains patter and matching string name W is actually started from index1 which means the 2nd position of the list. It is the same for the next table. All the position 0 of these lists are character ‘!’

4.3 Style Guidelines

My program follows the general indentation style of python 2.7. Every function within the class have been followed with comments. My variables were named by its definition and usage separated by a underline.

4.4 Innovations

I don’t think there exists some innovative points because I just design it following the general kmp algorithm

4.5 Terminal Session Script and Source Code

