DFS LAB – ASSIGNMENT 4

MTech(CS) I year 2018–2019

Deadline: 04 December, 2018

Total: 60 marks

SUBMISSION INSTRUCTIONS

1. Naming convention for your programs: `cs18xx-assign3-progy.c`

   IMPORTANT: Insert a single alpha-numeric string of your choice, 6-8 characters long, in the name given above as shown in the examples below. Think of this string as something like a security password, except that you are not required to remember the string. Examples: `cs1840-assign3-x19jdh4-prog1.c`, `ppo03wws-cs1840-assign3-prog2.c`, `cs1840-assign3-prog2-jsiwm7de.c`

2. To submit a file, go to the directory containing the file and run the following command from your account on the server (IP address: 192.168.64.35)

   ```
   cp -p name-of-your-file ~dfslab/2018/assign3/cs18xx/
   ```

   If you want to submit your files from a computer with a different IP address, run

   ```
   scp -p name-of-your-file \
   mtc18xx@www.isical.ac.in:/user1/perm/pdslab/2018/assign3/cs18xx/
   ```
   (enter your password when prompted).

   To submit all `.c` and `.h` files at one go, use
   `cp -p *.c *.h ~dfslab/2018/assign3/cs18xx/` or similar.

NOTE: Unless otherwise specified, all programs should take the required input from stdin, and print the desired output to stdout.

Q1. Let $S_1, S_2, \ldots, S_M$ be M stacks, with capacities $k_1, k_2, \ldots, k_M$, respectively. Suppose that, initially, all stacks are full, and no further PUSH operations can be done. The objective is to obtain a non-empty subset (not necessarily proper) of the stacks such that all stacks in the subset have the same number of elements. In order to achieve this objective, you may use as many POP operations as you like. Write a program that selects a subset such that the total number of elements in this subset is the maximum possible. If the total number of elements in multiple such subsets is the same, you should choose the subset that has fewer stacks. [10]

Input format: The capacities $k_1, k_2, \ldots, k_M$ will be provided as command-line arguments in order.

Output format: The indices of the stacks that are included in the selected subset, and the number of elements per stack in this subset. Note that stacks are indexed starting from 1.

Sample input 0: `./prog1 1 1 1 1`

Sample output 0: `1 2 3 4 1`

Sample input 1: `./prog1 8 1 4 2`

Sample output 1: `1 8`

A subset containing 8 elements in all can also be obtained by selecting stacks 1 and 3, and popping 4 elements from stack 1. However, this subset contains two stacks, rather than only one.

Sample input 2: `./prog1 100 80 150 120`

Sample output 2: `1 2 3 4 80`

Q2. Write an efficient program that takes a text file, and a list of strings, and for each string, prints the number of words in the text file that start with the given string. You may assume that the two files are small enough to fit in memory simultaneously, but do not make any other assumptions about the maximum length of words or input strings. [25]

Input format: Two file names will be provided as command-line arguments. The first file will contain plain English text (letters, digits, whitespace and punctuation marks). The second file will contain a list of alphanumeric strings, one per line. For this problem, you may assume that a word is defined as any non-empty sequence of letters and digits (not containing any other symbols). Note that case is **not significant** for this problem.

Output format: Your program should print on standard output one line corresponding to each alphanumeric string in the second input file. Each output line should contain two columns: the given string, and the number of words in the first file starting with that given string.

Sample input 0: `./prog2 input.txt strings.txt`

Contents of `input.txt`

```
All work and no play make Anand a dull man.
```

Contents of `strings.txt`

```
a
an
ma
e
```

Sample output 0:

```
a 4
an 2
ma 2
e 0
```

Q3. Your task in this problem is to implement and measure the performance of a *Bloom filter*, a data structure used for inexact searching in sets. Suppose S is a set stored using a Bloom filter. In response to a search for an element x, the Bloom filter returns one of two answers: x may be in S, or x is definitely not in S.

You may see `https://en.wikipedia.org/wiki/Bloom_filter` to get a preliminary idea for now. A properly formulated version of this question should be up by tonight.

[25]