DFS LAB – ASSIGNMENT 2

MTech(CS) I year   2018–2019

Deadline: 03 October, 2018

Total: 60 marks

SUBMISSION INSTRUCTIONS

1. Naming convention for your programs: `cs18xx-assign2-progy.c`
2. To submit a file (say `cs18xx-assign2-progy.c`), 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 cs18xx-assign2-progy.c ~dfslab/2018/assign2/cs18xx/
   ```

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

   ```
   scp -p cs18xx-assign2-progy.c \
   mtc18xx@www.isical.ac.in:/user1/perm/pdslab/2018/assign2/cs18xx/
   ```
   (enter your password when prompted).

   To submit all `.c` and `.h` files at one go, use
   `cp -p *.c *.h ~dfslab/2018/assign2/cs18xx/` or similar.

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

Q1. (a) Implement a generic data structure SEQUENCE to hold a list of elements of any one of the following types: `int`, `float`, or null-terminated strings (`char *`) (all elements in a particular list will be of the same type). Your data structure should support the operations given below.

- `void get_element(SEQUENCE s, size_t i)`: prints the numerical value of the i-th element of the sequence `s`, if it exists. The function should print an error message if the i-th element does not exist.
  The numerical value of an integer n is n itself; the numerical value of a floating point number f is the integer nearest to f; the numerical value of a string s of alphanumeric characters is the sum of the ASCII values of all the characters contained in s. Note that the numerical value of an empty string is 0.
- `size_t length(SEQUENCE s)`: returns the number of elements in the sequence `s`.
- `void summation(SEQUENCE s)`: prints the sum of the numerical values of the elements in the sequence `s`. Please see above for the definition of numerical value for sequences of various types. For a sequence containing no elements, `summation(s)` should print 0.

(b) Write a program that takes N sequences as input (from stdin), and prints the sequence `s` for which `summation(s)` is the maximum.

Input format: A positive integer N, followed by N sequences, one per line. Each of these lines will begin with `i`, `f` or `s` to specify whether the sequence on that particular line consists of `int`, `float` or string elements. This single letter will be followed by a non-negative integer that specifies the number of elements in the sequence, which in turn will be followed by the elements of the sequence.

Q2. Consider an unsorted array $A[0 \dots n-1]$ containing n *distinct* integers. After the array is sorted, suppose that element $A[i]$ of the original array moves to $A[j_i]$. Assuming that $|i - j_i| \leq k$ for all $i = 0, 1, \dots, n-1$, write a program to sort the original array A *efficiently*. You may use upto $O(k)$ additional space to store a heap.

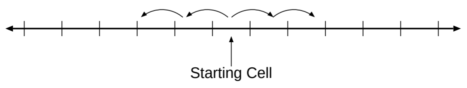
Input format: The value of k, followed by the elements of the array A will be provided as command line arguments.

Output format: The elements of the array A should be printed in sorted order.

Also write a program to sort the original array A using the `qsort()` library function. Insert appropriate calls to `getrusage()` in the programs to compare the times taken by `qsort()` and the sorting code in your program. **Do not** include the time taken for memory allocation, input, output or other parts of your programs. For each test case provided, run your program 10 times, and compute the average time taken to sort the input using the two methods mentioned above. Report the average times observed for each test case.

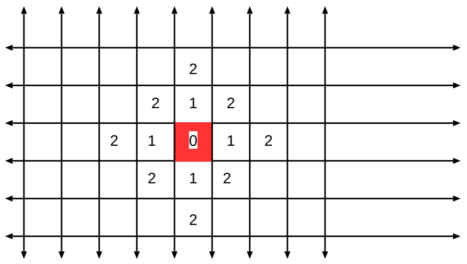
Q3. Write a program that takes a positive integer $n \leq 5$, and a non-negative integer $k \leq 100$ as command-line arguments, and computes the total number of cells reachable from any starting cell within an infinite, n-dimensional grid in k steps or less. See the examples below. You are only permitted to travel in a direction that is parallel to one of the grid lines; diagonal movement is not permitted.

NOTE: If you are interested, you may compute a closed-form expression for the required number, but **DO NOT** use the closed-form expression to compute the answer to this problem.



$n = 1, k = 2$

Number of cells reachable = 5



$n = 2, k = 2$

Number of cells reachable = 13

For $n = 3$, your output for $k = 0, 1$ and 2 should be 1, 7, and 25, respectively.