Institute for Advancing Intelligence (IAI), TCG-CREST
Mid-Semesteral Examination
Ph.D Program Session: 2022–2023
Subject: Introduction to Computer Programming and Data Structures

Date: 14. 10. 2022 Full Marks: 100 Time : 4 Hours

Note: Answer as much as you can. The maximum you can score is 100. Some of the questions requires files. They can be downloaded from digital version of the question paper kept in the *course webpage*. For submission, keep the names of the solution files as *MS0x_firstName.c* and send them to *laltu.sardar[at)outlook[dot)com* with subject as "*midsem_submission_firstName*".

1. Problem id #MS01: Count white-spaces.
   Take a set of filenames/filepaths from command line, count the number of appearance of the space ' ', tab '\t' and newline '\n' in each of the file and then output the counts for each of them.

   - Input: Sample input files are as follows.
     `input_MS01_1.txt`, `input_MS01_2.txt` and `input_MS01_3.txt`
   - Execution: You should run the code as `./a.out` followed by a set of file names separated by space. E.g., `./a.out input_MS01_1.txt input_MS01_2.txt input_MS01_3.txt`
   - Output format: `filename space_count tab_count newline_count` for each filename. E.g.,
     input_MS01_1.txt 252 32 12
     input_MS01_2.txt 12 23 34
     input_MS01_3.txt 34 45 3

   [20]

2. Problem id #MS02: Find and Replace.
   Consider three strings S_1, S_2 and S_3. Count all occurrences of S_2 in S_1, then replace them with the string S_3. Memory allocated for the output string should be full.

   - Input: Read the three strings from the 1^{st}, 2^{nd} and the 3^{rd} line of the file `input_MS02.txt`.
   - Output: Print the number of occurrences of S_2 and then print the processed string S_1 after the replacement operation in the next line.

   [20]

3. Problem id #MS03: Merit List.
   There is a student marks database `marks_file.txt` where there are some columns with the following information– student_id, attendance, marks_1, marks_2, marks_3, marks_4 and marks_5 separated by space. Another file `name_id_map.txt` that keeps names and ids separated by comma.

   (a) Construct a structure `Student` that stores all information relevant to a student. Then make an array of Student structures containing all information from above two files.

   (b) Construct a file with filename `meritList.txt` that contains only the name of students and average marks of best 4 marks, separated by comma so that the average marks are in sorted order.

   - Input: see file `marks_file.txt` and `name_id_map.txt`.
   - Output: a file `meritList.txt` in the following format
     $st\_id_1,\ avg\_marks_1$
     $st\_id_2,\ avg\_marks_2$
     $\vdots$
     $st\_id_n,\ avg\_marks_n$

[20]

4. Problem id #MS04: Linked List.

   (a) Create a linked list with the elements kept in the file `input_MS04.txt`.

   (b) Print the linked list after creation.

   (c) Given an integer p, remove the entries at the indices of multiples of p **from the end**.
   Example: Let the linked-list be the following with 11 elements
   477− > 912− > 51− > 951− > 517− > 64− > 678− > 14− > 198− > 697− > 213.
   Let $p = 3$. So the 3^{rd}, 6^{th} and 9^{th} element **from the end** need to be deleted. So the processed list should be following
   477− > 912− > 951− > 517− > 678− > 14− > 697− > 213.

   - Input: see file `input_MS04.txt`. Take p as run-time user input.
   - Output: Print the linked-list before and after the removal operation.

[20]

5. Problem id #MS05: Gambling.
   Suppose Adam plans to visit Goa with his friend Eve. They visit a casino and want to try their luck. They both started with Rs. 10,00,000 (Ten Lacs) coins. They simply choose a game of two dice. The game is as follows.

   A person first bets some amount of money on either the sum of dice less than 7 (smaller zone) or in greater than 7 (larger zone). Then the dealer rolls two unbiased dice. If the person wins the bet, it gets double of bet-amount, else the dealer keeps all money. Some of the pseudo-code is as follows.

- $choice\_of\_Adam \leftarrow$ `select_zone()`
- $choice\_of\_Eve \leftarrow$ `select_zone()`
- $bet\_amount\_Adam \leftarrow \geq$ 5% of Adam's total coin
- $bet\_amount\_Eve \leftarrow \geq$ 2% of Eve's total coin
- $dice\_sum \leftarrow$ `roll_die()`+`roll_die()`
- If $choice\_of\_Adam$ matches zone of $dice\_sum$, Adam's total coin is increased by bet_amount_Adam, else decreases that amount.
- If $choice\_of\_Eve$ matches the zone of $dice\_sum$, Eve's total coin is increased by bet_amount_Eve, else decreases that amount.

After playing 1000 rounds, find the coins Adam and Eve have.

- Bet amount must be an integer.
- `select_zone()` chooses either 0 or 1, which indicates smaller and larger zones respectively. `roll_die()` returns a random number between 1 and 6.
- A sample random number generating code is given `here`.
- **Output:** A file with 1000 lines where each line contains $round\_no$, $choice\_of\_Adam$, $choice\_of\_Eve$, $dice\_sum$, $value\_of\_Adam$, and $value\_of\_Eve$. They should be separated by tabs.

[20]

6. Problem ID # MS06: Deletion in a List.

Consider an integer list L of byte-length multiple of 5. There are two operations *insert* and *delete* as follows.

- $insert(L, x)$: inserts an integer x in the List. If L is already fill, at first extend the array keeping length multiple of 5.
- $delete(L, x)$: deletes all appearances of an integer x in the array L. After deletion, it rearranges the elements in the array. It reallocates the memory so that its length remains multiple of 5 not keeping more than 4 entries empty.

For the list, one can consider a contiguous integer array of length 5x or a linked list where each node stores an array of 5 integers. Dynamic memory allocation can be used for the same.

- Input: The number of inputs n followed by the inputs as
  n
  $op_1\ val_1$
  $op_2\ val_2$
  $\vdots\ \vdots$
  $op_n\ val_n$
  where $op_i \in \{+, -\}$ and val_is are positive integers.
  A sample input file `input_MS06.txt` can be downloaded.

- Output: input values followed by the elements in the list. E.g.,
  $op_1\ val_1 =>\ L[0],\ L[1], \ldots$
  $op_2\ val_2 =>\ L[0],\ L[1], \ldots$
  $\vdots$
  $op_n\ val_n =>\ L[0],\ L[1], \ldots$

[20]