tcg crest
Inventing Harmonious Future

Institute for Advancing Intelligence, TCG CREST
(TCG Centres for Research and Education in Science and Technology)

Introduction to Computer Programming and Data Structures
Assignments 03

Maximum Marks: 150
Clarification Deadline: 2023-Feb-28 Submission Deadline: **2023-Mar-02**

Assignment problem # AP0301

- Problem: Find the GCD of two integers using the Euclidean algorithm. Write two functions `GCD_R` and `GCD_I` with recursion and iteration, respectively. Compute the respective execution time $Time_R(a, b)$ and $Time_I(a, b)$ for input a and b, in microseconds.

- Input: n followed by N space separated a, b
  n
  a_1 b_1
  a_2 b_2
  $\vdots$
  a_n, b_n
  Where, $(a_i, b_i \in \mathbb{Z}$ and $0 < N \leq 10000)$. The name of the input file must be `input_AP0301.txt`

- Output: a_i b_i $gcd(a_i, b_i)$ separated by space
  a_1 b_1 $GCD\_R(a_1, b_1)$ $GCD\_I(a_1, b_1)$ $Time_R(a_1, b_1)$ $Time_I(a_1, b_1)$
  a_2 b_2 $GCD\_R(a_2, b_2)$ $GCD\_I(a_2, b_2)$ $Time_R(a_2, b_2)$ $Time_I(a_2, b_2)$
  $\vdots$
  a_n b_n $GCD\_R(a_n, b_n)$ $GCD\_I(a_n, b_n)$ $Time_R(a_n, b_n)$ $Time_I(a_n, b_n)$
  Time can be in second.

[30]

Assignment problem # AP0302

- *Building Rational number library.* A rational number is a type of real number, which is in the form of p/q where p, $q \in \mathbb{Z}$ and $q \neq 0$. A rational number can be represented as a structure of two integers– numerator and denominator.

```
struct rational {
int num;
int den; };
```

  Build your rational number library with the following operations.

  1. $0/1 \leftarrow$`rational_init` $(\&r, x, y)$, given an address of a rational numbers structure r and two values x, y, it outputs the numerator and denominator with x and y respectively. Finally returns 0 on failure and 1 on success.
  2. $r_3 \leftarrow$`rational_add` (r_1, r_2), given two rational numbers r_1 and r_2, it outputs another rational number r_3 such that $r_3 = r_1 + r_2$.
  3. $r_3 \leftarrow$`rational_sub` (r_1, r_2), given two rational numbers r_1 and r_2, it outputs another rational number r_3 such that $r_3 = r_1 - r_2$.
  4. $r_3 \leftarrow$`rational_mul` (r_1, r_2), given two rational numbers r_1 and r_2, it outputs another rational number r_3 such that $r_3 = r_1 * r_2$.
  5. $r_3 \leftarrow$`rational_div` (r_1, r_2), given two rational numbers r_1 and r_2, it outputs another rational number r_3 such that $r_3 = r_1/r_2$.

- Input: n followed by N space separated x_i, y_i, op_i, x'_i, y'_i
  n
  $x_1\ y_1\ op_1\ x'_1\ y'_1$
  $x_2\ y_2\ op_2\ x'_2\ y'_2$
  $\vdots$
  $x_n\ y_n\ op_n\ x'_n\ y'_n$
  Here op_i is one of $\{+, -, *, /\}$, $r_i = x_i/y_i$, $r'_i = x'_i/y'_i \in \mathbb{Q}$. Input file `input_AP0303.txt`

- Output: If $r''_i = x''_i/y''_i$ is the output of ith input then
  $x''_1\ y''_1$
  $x''_2\ y''_2$
  $\vdots$
  $x''_n\ y''_n$

[40]

- A rational number p/q is said to be in canonical form if p and q are co-prime, and $q > 0$. Output the results of the above four operations in canonical form. Hint: use GCD..

[10]

Assignment problem # AP0303

- Matrix Library: Consider you are building a library for matrix operations. Let the file be "myMatrix.c". The file should have the following functions.

  1. `float ** my_malloc_2D_float(int n, int m)`: It takes dimension of any 2D matrices $A_{n\times m}$ and allocates memory for that matrix and returns the pointer to the allocated memory as `float **`. It returns `NULL` in case of failure.
  2. `void show_2D_matrix_float(float **A, int n, int m)`: It takes a 2D array pointer and its dimension. It prints the elements of the matrix $A_{n\times m}$. It prints "show_2D_matrix_float: failure to print matrix", in case of failure.
  3. `scan_2D_f_matrix_from_opened_file(float **A, int n, int m, FILE * inp_file_ptr)`: It takes a 2D array pointer, its dimension. It scans the elements of the matrix $A_{n\times m}$ from the file opened already in inp_file_ptr. It returns 0 in case of failure. It also prints a failure message in the terminal before return.
  4. `scan_2D_f_matrix_from_file(float **A, char inp_file_name[] )`: It takes a 2D array pointer. First, it scans the dimensions m, n of the matrix, then it scans the elements of the matrix $A_{n\times m}$ from the file inp_file_name. It returns 0 in case of failure. It also prints a failure message in the terminal before return.
  5. `float ** matrix_mult_f(A, B, m, n ,p)`: This takes 2 float matrices and outputs its products in a new matrix. It returns NULL in case of failure. It also prints a failure message in the terminal before return.
  6. `void matrix_free(A, m, n)`: This takes a matrix and free the memory allocated to the matrix. Hint. first it frees the memories of each rows and then free the array that stores the address of the rows. Your program should not be terminated. In case of any error, just return the error message.
  7. Write your own main function that tests matrix multiplication as follows.
     (a) Suppose the program file `myMatrix.c` is compiled. Then run as
     `./a.out input_matrix_a.txt input_matrix_b.txt .`
     Here "input_matrix_a.txt" and "input_matrix_b.txt" are two files that store 2D matrices. The program should output the product of the two matrices kept in "input_matrix_a.txt" and "input_matrix_b.txt". Thus, for multiplication, pass the name of the matrix files from the command line.
     (b) The main function first reads the first file, scans dimensions of the matrix, allocates memory for that matrix, scan the matrix entries say in A. Then it does the same for the next file, say in B.
     (c) Then it calls `void ** matrix_mult(A, B, m, n ,p)` to multiply the matrices and prints the result matrix in the terminal.
     (d) Try to return an error message in case of any failure.

[70]