BİL 102 – Computer Programming
HW #9

Last Submission Date: April 29, 2015 – 11:00

For Submission and/or Questions about the HomeWork # 9 see Ar. Gör. Betül Türkkol (Room 108-Computer Vision Lab)

1. You will write a C program for a university. A person can be one of following two types:

(I) Instructor

(E) Employee

write 2 structure types for describing a person

type_I should have 3 strings for name, surname, department, 2 classes that he/she has to teach, one double for salary.

type_E should have 2 strings for name, surname, a double for salary, and a char to store degree of employment (**degree a :** shows the employee is working as head of the university; **degree b :** shows the employee is working as a department secretary; **degree c :** shows the employee is working for general service of the university such as canteen, cleaning etc.)

combine_type should have a **char** to decide structure type ('I' or 'E') and a **union** that has 2 different types of person.

Write a function to calculate annual salary increase of a person in the university. The function takes a person information and returns updates person information.

- If the person is an Instructor, increase the salary 5% for first-half of the year and 5.3% for the second-half.
- If the person is an Employee, the salary is increased such that
  - 17.5% for "**degree a"** employees
  - 12% for **"degree b"** employees
  - 9% for **"degree c"** employees.

combine_type salary_rise(combine_type person_info);

Write a function that takes person information from an input file update the salary information (calculate annual salary increase) and write into a binary file. The names of the text and binary files are parameters to the function. An example text file is as follows:

Instructor, John Parker, Computer Science, CSE100, CSE200, 100000
Employee, Mary James, 300000, a
Employee, Terry Maple, 35000, b
Instructor, Ali Topuz, Biology, BIO244, BIO120, 80000
Employee, Carol Heinz, 40000, b

2. Consider the implementation of **Operators of Complex Numbers** in the text book **(pg 569)**; using the program you will create a library called **complex** with an implementation file **complex.c** and a header file **complex.h**. Test your library with an application.

3. Microbiologists estimating the number of bacteria in a sample that contains bacteria that do not grow well on solid media may use a statistical technique called the most probable number (MPN) method. Each of five tubes of nutrient medium receives 10 ml of sample. A second set of five tubes receives 1 ml of sample per tube, and in each third set of five tubes, only 0.1 ml of sample is placed. Each tube in which bacterial growth is observed is recorded as a positive, and the numbers for the three groups are combined to create a triplet such as 5-2-1, which means that all five tubes receiving 10 ml of sample show bacterial growth, only two tubes in the 1 ml group show growth, and only one of the 0.1 ml group is positive. A microbiologist would use this combination of positives triplet as an index to table like the table below to determine that the most probable number of bacteria per 100 ml of the sample is 70, and 95% of the sample yielding this triplet contain between 30 and 210 bacteria per 100 ml.

Combination of Positives	**MPN index/100 ml**	**95% Confidence Limits**	
		Lower	**Upper**
4 – 2 - 0	22	9	56
4 – 2 - 1	26	12	65
4 – 3 - 0	27	12	67
4 – 3 - 1	33	15	77
4 – 4 - 0	34	16	80
5 – 0 - 0	23	9	86
5 – 0 - 1	30	10	110
5 – 0 - 2	40	20	140
5 – 1 - 0	30	10	120
5 – 1 - 1	50	20	150
5 – 1 - 2	60	30	180
5 – 2 - 0	50	20	170
5 – 2 - 1	70	30	210
5 – 2 - 2	90	40	250
5 – 3 - 0	80	30	250
5 – 3 - 1	110	40	300
5 – 3 - 2	140	60	360

Define a structure type called **Row** to represent one row of the MPN table. The structure will include one component for the combination-of-positives triplet (which is of type **triplet_t** including three integers) and three integer components in which to store the associated most probable number and the lower and upper bounds of the 95% confidence range. Write a program to implement the following algorithm for generating explanations of combination-of-positives triplets.

a. Consider the data in the table above stored in a .txt file. Write a function called Into_Binary that takes two file pointers; one is for input text file and the other is for a binary file called **converted.bin**; reads the data from text file and writes into binary file. **Note** that your function can read 10 lines at a time, hence you will have to carry your data by more than one read.

void Into_Binary(FILE *text_input, FILE *binary_output)

b. You will write a function called **Load_Mpn_table** that takes pointer to binary_input file an array **mpn_table**, and **maximum size** (10) to read data from the binary file that you have created in part a into an array that called **mpn_table**. Then it returns the actual size of array. **Note** that your function can read at most 10 records at a time, hence you will have to perform operation on your data by more than one read.

int Load_Mpn_table(FILE *binary_input_file,Row mpn_table[], int maxsize)

c. Repeatedly get from the user a combination-of-positives triplet, search for it in the combination_of_positives components of mpn_table, and generate a message.

For example; if user enter 5-2-1 as input; your function should print the message: **MPN=70; 95% of samples contain between 30 and 210 bacteria/ml**.

In order to achieve this goal you will need to write another function called **search** that takes the **mpn_table**, its **actual size** and a string representing a combination_of_positives triplet; and returns the subscript of the structure whose combination_of_positives component matches the target or **-1** if not found.

Row search(Row mpn_table[], int actual_size, const triplet_t triplet_to_search)

General:

1. Obey honor code principles.
2. **Read your homework <u>carefully</u>** and follow the directives about the I/O format (data file names, file formats, etc.) and submission format **<u>strictly</u>**. Violating any of these directives will be penalized.
3. Obey coding convention.
4. Your submission should include the following files **<u>and NOTHING MORE</u>** (no data files, object files, etc):
   HW09_<student_name>_<studentSirname>_<student number>_part1.c
   HW09_<student_name>_<studentSirname>_<student number>_part2.c
   HW09_<student_name>_<studentSirname>_<student number>_part3.c
5. Do not use non-English characters in any part of your homework (in body, **file name**, etc.).
6. Deliver the printout of your work **until the last submission date**.