Final Project Instructions
C Programming for Engineer

(Take Home Exam – 2018/2019)

Further Particulars

(1). Answers must be typed in English; you should include any codes, diagrams, illustrations, tables and figures that you may have used to support your answers.
(2). List any references you used in your work, if any, in *reference section.*
(3). Name your *.c / *.h source code according to the respected question (question1.c and question1.h for Question 1, question2.c for Question 2, etc)
(4). Save your screenshots showing the program output in jpeg/png format, and embed them in your report.
(5). Save your report in **pdf** file, and along with any supported files (*.c and *.h files), zip them and submit via email to `ipg@ieee.org` using email subject:
"**[FINAL-CPROG]** ⟨**student_number**⟩ ⟨**your_name**⟩"
(note the use of the square bracket for subject heading).
(6). File name format:
FINAL_CPROG_nim_name.pdf for the report, and
FINAL_CPROG_nim_name.ZIP for the zip file
(7). Deadline: Thursday, 24 December 2018.
(8). There are 4 Tasks in this Exam with a total mark of 100.
(9). Note that there will be penalties for submission that do not adhere to this instructions.
(10). Any forms of cheating including, but not limited to, plagiarism or attempts of it is a serious offense, and it will lead to your work dismissal.

TASKS

Taks 1. *(point: 30)*
Write a C program that asks the user for an integer number, n, which is less than 1000, i.e., $n < 1000$, and then prints (on the standard output screen) the number from 1 to n. However, for multiples of three, instead of printing the number, the program will print "Happy". Similarly, for the multiples of five, it will print "New Year", and for numbers which are multiples of both three and five, the program will print "Happy New Year!!!"

When the program finishes printing out the number on the screen, it will write the following line of sentence to a text file:

> We have printed x Happy, y New Year, and z Happy New Year.

with x, y, and z are the numbers of occurrences of multiples of three, five, and both three and five, respectively.

The program should save the text file as happy.txt every time it is running, and append new line of the above sentence for the subsequent run of the program.

Following is a sample output after the program is run:

```
Enter a number [1-1000]: 2000 [enter]
Invalid input! Try again ...? 20 [enter]

1
2
Happy
4
New Year
Happy
7
8
Happy
New Year
11
Happy
13
14
Happy New Year
16
17
Happy
19
New Year

[The program then will write the sentence
``We have printed 5 Happy, 3 New Year, and 1 Happy New Year.''
in the text file]

Try again (y/n)? y [enter]

Enter a number [1-1000]: 6 [enter]

1
2
Happy
4
New Year
Happy

[The program then will write the sentence
``We have printed 2 Happy, 1 New Year, and 0 Happy New Year.''
in the text file]

Try again (y/n)? n [enter]

Thank you for using this program!

[program ends]
```

After the program running, the text file happy.txt will contain the followings:

```
We have printed 5 Happy, 3 New Year, and 1 Happy New Year.
We have printed 2 Happy, 1 New Year, and 0 Happy New Year.
```

Taks 2. *(point: 40)*

The following C code declares variable *mtx* of matrix type, and dynamically allocates memory for *mtx*. It will then get keyboard input for N-by-N matrix, computes/prints the sum of all elements in the matrix. Subsequently, the program deallocates (free) the memory of the matrix.

```
1
2     #define N 5
3     int main(){
4       float** mtx;
5       float mtx_sum = 0.0;
6       int i , j;
7
8       //
9       // In this part insert code for
10      // dynamic memory allocation for mtx (N-by-N matrix)
11      //
12      //
13      //
14
15      for (i = 0 ; i < N ; i++) {
16        for (j = 0 ; j < N ; j++) {
17          scanf("%f",&mtx[i][j]); mtx_sum += mtx[i][j];
18        }
19      }
20
21      printf("sum of all matrix elements : %f\n",mtx_sum);
22
23      //
24      // In this part insert code for
25      // dynamic memory deallocation (free)
26      // for mtx (N-by-N matrix)
27      //
28      //
29
30      return 0;
31    }
32
```

A. Your task is to write the codes to dynamically allocate and de-allocate the memory for the matrix.

B. Subsequently, you have to modify the above code so that now the program will first ask the user to enter the size of the matrix, accept two matrices of the same size (maximum size 5-by-5) where each element of these matrices are manually input, and then perform matrix multiplication. The program will then output all the input matrices and the multiplication results, both to the standard output screen and also to the text file called matrix.txt.

An example of the output on screen as well as in the matrix.txt text file are as follows:

```
Matrix size = 2-by-2
First matrix =
1 0
0 1

Second matrix =
1 2
3 4

Multiplication result =
1 2
3 4
```

Taks 3. *(point: 20)*

Your task is to implement a function **epowerx** that has an input parameter x of floating-point value (real numbers), and then return the approximation of e^x as computed using the formula

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^{10}}{10!}.$$

Use this function in a C main program that asks users for the x value, perform the calculation, and output the result.

To assist the implementation, you may use the recursive function $fact(n)$ and $power(x, n)$ given below.

```
1
2     float epowerx(float x){
3       //
4       // This is where you implement the function
5       //
6     }
7
8     // Function to compute factorial
9     int fact(int n){
10      // computes and returns n!
11      if (n==1 || n==0) return 1;
12      return n*fact(n-1);
13    }
14
15    // Function to compute power
16    float power(float x, int n){
17      // computes and returns
18      if (n==0) return 1;
19      return x*power(x,n-1);
20    }
21
```

Taks 4. *(point: 10)*

The following code is supposed to perform a certain calculation, but generates a wrong result. What do you thing the code is trying to calculate? Explain what is wrong and why? Your task is to modify the code to correct it!

```
1   #include <stdio.h>
2   int fact(int n);
3   double compute_e();
4   int main(){
5     printf("%lf\n" , compute_e());
6     return 0;
7   }
8
9   int fact(int n){
10    int i, product = 1;
11    for (i = 1 ; i<= n ; i++) product *= i;
12    return product;
13  }
14
15  double compute_e()
16  {
17    int i;
18    double sum = 0.0;
19    for (i = 0 ; i <= 8 ; i++) sum += 1 / fact(i);
20    return sum;
21  }
22
```

GOOD LUCK!