Final Exam 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, if any, 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 (i.e., question1.c and question1.h for Question 1, question2.c and question2.h for Question 2, etc). Put your name (and student numbers, if you have one) in the comment lines of your source codes.
(4). PUT COMMENTS in your codes to explain in plain English what you do.
(5). Save your screenshots showing the program output in jpeg/png format.
(6). Save all your works in one ZIP document and send to `ipg@ieee.org` using email subject "[FINAL-CPROG-19] ⟨student_number⟩ ⟨your_name⟩" (note the use of the square bracket for subject heading).
(7). File name format for the zip file:
FINAL_CPROG_nim_name.ZIP
(8). Deadline: Wednesday, 21 August 2019, 09.00 WIB.
(9). There are 3 Tasks in this Exam with a total mark of 100.
(10). Note that there will be penalties for submission that do not adhere to this instructions.
(11). 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: 20)*
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
```

⟹ (Next task) . . .

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 the N-by-N 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]);
18        }
19      }
20
21      //
22      // In this part insert code for
23      // dynamic memory deallocation (free)
24      // for mtx (N-by-N matrix)
25      //
26      //
27
28      return 0;
29    }
30
```

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 addition. The program will then output all the input matrices and the matrix addition 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

Addition result =
2 2
3 5
```

$\Longrightarrow$ (Next task) ...

Taks 3. *(point: 40)*

Your task is to implement a function **edoublepowerx** that has an input parameter x of floating-point value (real numbers), and then return the approximation of e^{2x} as computed using the formula

$$e^{2x} = 1 + \frac{2\,x}{1!} + \frac{4\,x^2}{2!} + \frac{8\,x^3}{3!} + \ldots + \frac{1024\,x^{10}}{10!}.$$

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
```

GOOD LUCK!