COMPRO1 (for S18A and S19A) MP Specifications – Part 1 of 3
Taylor Series Expansion of Sine and Cosine Trigonometric Functions
Part 1 is 20% of the MP Grade

I. INTRODUCTION

The 1st part of the machine problem is concerned with the (approximation of the) trigonometric functions **cos *x*** and **sin *x*** (cosine and sine) where ***x*** is a parameter representing the angle in radians.*

The Taylor series expansions of these trigonometric functions are:

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

The exclamation mark (!) denotes the factorial symbol. Note that the ellipses indicate that there maybe more terms to follow in the series. In the equations shown above, the approximation explicitly shows only 4 terms for each function. In the case of the cosine function, the first term is 1, the second term is $\frac{-x^2}{2!}$, the third term is $\frac{x^4}{4!}$, and the fourth term is $\frac{-x^6}{6!}$. Note that increasing the number of terms improves the accuracy of the approximation.

The Taylor series shown above is quite interesting from the viewpoint of programming. To implement the series, the programmer should apply his/her knowledge of:

- mathematical expressions
- conditional control structures
- counters
- accumulators, and
- loop control structures

The objective, therefore, of this machine problem is for you to demonstrate your knowledge of these concepts.

II. REQUIREMENTS

Your task is to write a C program that computes the Taylor series approximation of the **cos *x*** and **sin *x*.** The angle *x* is double data type.

For Part I of the machine problem, you are required to write/implement all codes using `main()` only. DO NOT define any other function aside from `main()`.

* It is assumed that you are familiar with the concepts of these trigonometric functions since they should have been covered as part of your mathematics subjects in your high school or first year college years.

You are allowed to use only standard C99 programming concepts/constructs discussed in class (and in the handouts). Non-standard C constructs (for example, the **`clrscr()`** or **`gotoxy()`** functions) are prohibited; usage of such constructs will automatically result into a grade of 0.0 for this particular requirement.

The program DOES NOT require any input from the user.

The output of the program is a tabular list of values of the angles in both degrees and radian units, and the corresponding cosine and sine of the angles. Note that your program should convert angles from degrees to radians.

To test the computations for both positive and negative angles, approximate the cosine and sine from -180 degrees to 180 degrees in increments of 5 degrees. The program's output should have the following format. The values in the first column shows the angle in degrees, the second column is the angle in degrees, the third column is for the cosine of the angle and the last column is for the sine of the angle.

```
<your IDNUMBER>
<one empty line>
-180.00        <value>          <value>   <value>
-175.00           :                :         :
-170.00           :                :         :
    :             :                :         :
    :             :                :         :
    :             :                :         :
-10.00            :                :         :
-5.00             :                :         :
0.00           <value>          <value>   <value>
5.00              :                :         :
10.00             :                :         :
    :             :                :         :
    :             :                :         :
    :             :                :         :
170.00            :                :         :
175.00            :                :         :
180.00         <value>          <value>   <value>
```

Indicate 2 digits after the decimal point for the value of angle x in degrees, and 12 digits after the decimal point for the values of x in radians, and for the corresponding cosine and sine values.

It is imperative that you follow the format because the result of your program will be checked automatically using another program. Point deductions will be applied for not following the format.

Three example results (using different number of terms) will be supplied to you together with this document for your reference.

IMPORTANT NOTE #1:
HONESTY POLICY APPLIES. This means that you are expected to produce/and submit a project that you DID on your own, and not by other people. Conversely, it means that you DID NOT create the work submitted by another student. **Violation of this policy will result into a FINAL GRADE of 0.0.**

If you have any question/clarification regarding the problem, please feel free to approach or email me.

IMPORTANT NOTE #2:
Include a **PREAMBLE** in the form of comments in the first few lines of your C source program indicating the following for identification purposes

```
/*  COMPRO1 Machine Problem Part I                          */
/*  <LASTNAME>, <FIRSTNAME>                                 */
/*  <SECTION>                                               */
/*  <short description of what this program is all about */
```

IMPORTANT NOTE #3:
Include the following macro definitions in your program:

```
#define PI   3.141592653589
#define NUMBER_OF_TERMS    10
```

The macro **PI** indicates the value of math constant π (PI) which you will need to convert the angles from degrees to radian units. The macro **NUMBER_OF_TERMS** indicates the number of terms to be used in the approximation. You need to use this macro in your program. Later on, you might want to try changing it to other values, for example, 5 or 10 or 15 to see the improvement in the approximation. However, before you submit your source code, make sure that **NUMBER_OF_TERMS** is set to **10** (not some other value).

The only header file that should be included is <stdio.h>. Other header files are NOT allowed.

III. DELIVERABLES AND SUBMISSION DEADLINES

You need to submit THREE items, namely:

1. Your C program source code. Use your last name, followed by an underscore, followed by mp1 as filename. For example, if your last name is SANTOS, your file should be named **santos_mp1.c**.

   The softcopy of your source code should be **RECEIVED** as an email attachment file **before 11:59PM of July 21, 2014 (Monday)**. Send your email to the **TWO** addresses indicated below:

   florante.salvador@dlsu.edu.ph
   pulingfe@yahoo.com

   Email subject should be:

   COMPRO1 LASTNAME<underscore>FIRSTNAME <space> SECTION<space>MP1

2. Accomplished MP submission checklist (print and accomplish the document named "mp1_checklist.pdf").

3. On short-bond paper, the hardcopy/printout of your:
   a. C program source code
   b. 3 sample output corresponding to **NUMBER_OF_TERMS** equal to 5, 10 and 15 respectively.

Staple the MP submission checklist on top the program and sample output hardcopies. Submit these **on July 22, 2014 (Tue)** within the **first 10 minutes of your own COMPRO1 class hours only** (not those of another section). Note that the softcopy of the source code must be exactly the same as the hardcopy. That is, DO NOT modify your program after submitting it via email.

INCOMPLETE SUBMISSION (i.e., did not email the source code, or did not submit the checklist, or did not submit the hardcopy of source code or sample output) will automatically result into a grade of 0.0 for this particular requirement.

LATE SUBMISSION WILL **NOT BE ACCEPTED** unless there is a valid and verifiable excuse (example: sickness, emergency).

IV. TESTING, CHECKING and GRADING SCHEME

Your source code will be compiled, linked and tested. **BLACK-BOX TESTING will be used to check your submission.**

Note that by default you will receive a perfect grade of 100%. It will remain 100% if there are no programming errors and that you complied properly with all the specifications.

Deductions will be applied based on the following:

- Programming error – syntax error and/or logical error will result into a major deduction; see table below
- Non-compliance with specification – will result to a 2% minor deduction **for each** infraction; for example: incorrect filename, incorrect printout, etc.
- Bad programming style – will result to a 10% minor deduction for source codes that DO NOT follow recommended "good" programming practices thus resulting to difficult-to-read codes due for example to incomprehensible variable name, bad indentation, etc.

A simple grading scheme (no minor deduction applied yet) is indicated in the following table.

Grade	Remarks
100%	Correct results
50%	Half-correct only (for example: cosine is correct but sine is not
10%	Error/s in compilation, linking or execution (logical error) *(note that 10% is essentially just a "consolation" point)*

As mentioned above, the output of your program will be checked automatically using another program. I will be browsing your program code but will not try to check it in detail for semantics nor point out the error if there is something wrong with your solution. In case your program turns out NOT to be 100% semantically correct, you still have a lot of chance to fix it in Part 2 of the machine problem.

V. REFERENCE

Check out the following site for Taylor series expansions of other related trigonometric functions.

`http://www.efunda.com/math/taylor_series/trig.cfm`

***** End of the Machine Problem Specifications *****

Last Note: Consult me immediately if you have any clarification, question or problem regarding this academic activity. Enjoy! ☺☺☺