Weekend Exercise

"Do or do not; there is no try"

-Jedi Master Yoda (The Empire Strikes Back)

Instructions:

(a) In this exercise set, please **PRACTICE** writing down your answers without the use of a C compiler (or its variants). The rationale behind this is that we want to learn to write programs that are correct and conforming to C language's syntax, without relying on the compiler to tell us what language rules we've violated[1]. Once you have your solution, to **validate** if you have done correctly and produce the expected output, kindly encode your codes using IDE or any text editor, compile and run using **gcc** [*test your program under different input values.*]

Even though, in the end, we will still encode your solution using an IDE, the **idea** here is to organize your thoughts, form your codes and eventually produce your solution ***first*** using paper and pen; able to understand, analyze codes without using any compiler ☺

(b) Please remember on how C compiler handles mathematical computations dealing with numbers that are either represented as integers or real numbers (i.e., floating point). Also be conscious when different data types are used in one statement!!

(c) Before writing your program, analyze the given problem by identifying:

(i) What are the input values supplied (if any) or what are those input values that need to be taken from the user? In this process, ***reflect and discern*** what would be the appropriate data type to represent those input data. For example, when computing the appropriate tax to be paid by any working individuals, one of the items being asked and considered in tax calculation is the number of dependents if any, by the individual. If you are asked to define a variable (numDependents) corresponding to this information, what data type would best to represent numDependents? – will it be integer? float? Or char?

(ii) What are the desired computations or what does the problem wishes us to do? And

(iii) What are the desired outputs or data that need to get printed? Similar to (i), you need to ***reflect and discern*** the appropriate data representation (i.e, output should be in float form? integer form? char form?) when performing output-related actions.

A good understanding of the problem will surely result to a good planning, organization and development of solution ☺

(d) Try to ***remember and understand*** all the learning you will get from this exercise set as some of these might be your questions later on in one of our lectures or activities in class, which I may not provide opportunity to answer them. There are things (at least those listed in the syllabus) and other programming-issues that I would prioritize to discuss ☺

(e) **Suggestion:** Best to print out a copy of this exercise set, and write down any observations, answers, and realizations you may have on this activity.

==========

[1] During the departmental exams (and even final exam) your programming skills will be tested using paper and pen ☺

Exploration, Observation, Reflection…. Why?:[2]

Before typing the following short programs using an IDE (or any text editor), ***deduce*** if the programs given are syntactically AND logically correct **[let us use first your brain powers☺]**. If it is, identify what will ***get printed on screen*** if the program is executed; Otherwise (this means that programs have errors – either logical or syntactical), identify ***what are the errors*** and ***explain*** the reason of errors.

Program 1:

```
#include <stdio.h>
int main(){
  int nValue1, nValue2=10;
  printf("Enter a value: ");
  scanf(%d, %nValue1); //[3]
  printf("Value1 is:%d\n",nValue1);
  return 0;
}
```

Program 2:

```
#include <stdio.h>
int main(){
 int nValue1, nValue2=10;
 printf("Enter a value: ");
 scanf("%d", &nValue1);          //[4]
 printf("Value1 read is: %d\n", nValue1);
 scanf("%d", &nValue1);          //[5]
 printf("Value1 now:%d\n",nValue1);
 printf("Value2 now:%d\n",nValue1+nValue2);
 printf("Value1 now:%d\n",nValue1);
 printf("Value2 now:%d\n",nValue2);
 return 0;
}
```

The two programs (Program 1 and Program 2) below are nearly similar, except the difference is on printing the values. We want to see the effect of printing integer-assigned variables following floating-point format (Program 2) – assuming Program 2 is valid ☺. Is it valid? If it is, what makes you think the compiler produced the output in Program 2 different from Program 1?

Program 1:

```
#include <stdio.h>
int main(){
  int nValue1, nValue2=10;
  nValue1 = nValue2*10*10;
  nValue2 = nValue2 +1;
  printf("Value1 is: %d \n", nValue1);
  printf("Value2 is: %d \n", nValue2);
  return 0;
}
```

Program 2:

```
#include <stdio.h>
int main(){
  int nValue1, nValue2=10;
  nValue1 = nValue2*10*10;
  nValue2 = nValue2 +1;
  printf("Value1 is: %f \n", nValue1);
  printf("Value2 is: %f \n", nValue2);
  return 0;
}
```

In the following programs, follow the general instructions stated above ☺

Program 1:

```
#include <stdio.h>
int main(){
  float This_One =2;
  float That_One;
  That_one = 2 * This_one;
  printf("Value is %f\n", That_one);
  return 0;
}
```

Program 5:

```
#include <stdio.h>
int main(){
  float This_One =2
  float That_One;
  That_One = 2 * This_One;
  printf("Value is %f\n", That_One);
  return 0;
}
```

Program 2:

```
#include <stdio.h>
int main(){
  float c_Value1 = 2; /* Value 1  */
  float c_Value2;     /* Value 2
  c_Value2 = c_Value1 + c_Value1;
  printf("Value is %f\n", c_Value2);
  return 0;
}
```

Program 6:

```
#include <stdio.h>
int main(){
  float num=125.738;
  printf("A notation on 125.738:%f\n", num);
  printf("Another notation: %d\n", num);
  printf("Another notation: %e\n", num);

  return 0;
}
```

[2] Please note that I didn't place the library <conio.h> and use of getch() function since you will analyze these codes initially without the assistance of a C compiler ☺

[3] Assume the user entered a value 100 at this point

[4] Assume the user entered a value 100 at this point

[5] Assume the user entered a value of 50 at this point

Program 3:
```
#include <stdio.h>
int main(){
  float ThisValue = 2; /* Value 1 */
  float ThatValue;     /* Value 2 */
  ThatValue = thisValue + thisValue;
  printf("Value is %f\n", ThatValue);
  return 0;
}
```

Program 7:
```
#include <stdio.h>
int main(){
  float ThisValue = 15;    /* Value 1 */
  float ThatValue;         /* Value 2 */
  ThatValue = ThisValue + ThisValue;
  printf("Value is %f\n", ThatValue);
  return 0;}
```

Program 4:
```
#include <stdio.h>
int main(){
  float ThisValue = 2;  /* Value 1 */
  float ThatValue;      /* Value 2 */
  ThatValue = ThisValue + ThisValue
  printf("Value is %f\n", ThatValue);
  return 0;
}
```

***For curiosity's sake**: The following programs are nearly similar to one another, the goal here is to determine the outcome/result if data is read and printed other than its defined data type. Try to figure out what will be received in scanf() statement and what will get printed in printf() statement. Use C compiler to validate your "guess".*[6]

Program 1:
```
#include <stdio.h>
int main(){
  float nValue1;
  printf("Enter a value:");
  scanf("%f", &nValue1);
  printf("Value1 is:%f \n", nValue1);
  return 0;
}
```

Program 3:
```
#include <stdio.h>
int main(){
  float nValue1;
  printf("Enter a value:");
  scanf("%d", &nValue1);
  printf("Value1 is:%f \n", nValue1);
  printf("Value1 is:%d \n", nValue1);
  return 0;
}
```

Program 2:
```
#include <stdio.h>
int main(){
  int nValue1;
  printf("Enter a value:");
  scanf("%f", &nValue1);
  printf("Value1 is:%d \n", nValue1);
  return 0;
}
```

Program 4:
```
#include <stdio.h>
int main(){
  int nValue1;
  printf("Enter a value:");
  scanf("%f", &nValue1);
  printf("Value1 is:%f \n", nValue1);
  return 0;
}
```

***For curiosity's sake**. Try the following program using a C compiler and see what will get printed. Study what each format string corresponds to.*

Program 1:
```
#include <stdio.h>
int main(){
  printf("Value is %d\n", 15);
  printf("Value is %i\n", 15);
  printf("Value is %x\n", 15);
  printf("Value is %o\n", 15);
  return 0;
}
```

Question: What is the different about the outputs?

Program 3:
```
#include <stdio.h>
int main(){
  printf("Value is %f\n", 15.0);
  printf("Value is %e\n", 15.0);
  printf("Value is %E\n", 15.0);
  return 0;
}
```

Program 2:
```
#include <stdio.h>
int main(){
  char letter;
  letter = 'b';
  printf("Value is %c\n", letter);
  printf("Value is %d\n", letter);
  printf("Value is %x\n", letter);

 return 0;
}
```

[6] You can do similar observations in your free time dealing with other basic data types ☺

MAGIC 8 QUESTIONS: Answer the following:

1. What is the purpose of /* */?
2. What indicates the beginning and the end of program instructions in the C language?
3. What marks the point where the C program begins execution?
4. What tells the C compiler to include the standard input/output file?
5. What is it called when you combine a variable declaration with an assignment operator?
6. State the unique characteristic of integer division in C.
7. What is the purpose of scanf() function?
8. What is the purpose of printf() function?

MAGIC 8 – PROGRAMMING ☺ Let's try to write program!

1. Write a C program that will compute an employee's **gross pay**, **withholding tax** (that is 15% of the gross pay) and the **take-home pay** (that is gross pay less of the withholding tax) basing on employee's hourly rate and number of hours worked. The gross pay is computed by multiplying the number of hours worked with his hourly rate. A sample scenario is: if Employee E1 earns 120.00php for every hour he works, and let us say, in a particular day, he works for a total of 8.5 hours. How much does he gets at the end of the day [i.e., what will be his gross pay, tax to be deducted from his gross pay and his take-home pay]?

2. Write a C program that given an integer representing the number of years, you are to print the number of hours it covers.

3. Write a C program that will accept a speed value in miles per hour. Your program should print the input value's equivalence in feet per second. For this problem, you might need to search at the Internet or browse through your Science notes for some relevant formula that will help you on the conversions.

4. Write a program that computes for the equivalent resistance (ER) for two resistors (r1 and r2) in parallel. The equivalent resistance is computed as:

$$ER = \frac{r1 * r2}{r1 + r2}$$

5. Write a C program that will solve for the power dissipation of a resistor when the voltage across the resistor and the current in the resistor are known. The relationship for resistor power dissipation is: P = IE, where P corresponds to the power dissipated in watts; I is the resistor current in amps; and E is the resistor voltage in volts.

6. Write a C program that will solve for the current in a series circuit consisting of three resistors and a voltage source. The program should require the user to input the value of each resistor and the value of the voltage source. The relationship for total current is defined below, where I is the total circuit current in amps; V is the voltage source in volts; and r1, r2 and r3 are values for each of the resistor in ohms:

$$I = \frac{V}{r1 + r2 + r3}$$

7. A friend of mine once taught me a magic number trick: any number I will give between 1 and 10, applying the number I picked with some equations will always result to 13. The algorithm/process of calculations he taught me is as follow:

a. Pick a number between 1 and 10. Let us say, for easy referencing, we call the number picked is X.
b. Multiply this number by 9.
c. Using the resulting product, add the digits together. For instance, if the resulting product is 10, we add 1 and 0. If the resulting product is a single digit, say 8, we will add 0 and 8.
d. Using the result in c above, we add it with 4.

e. The final answer is 13.

Your task is to **validate** if indeed the above algorithm works for all numbers from 1 to 10. So instead of **manually** calculating from 1 to 10 using the algorithm above, write a C program that executes the algorithm stated. Test your program at most 10 times -beginning with an input value 1, then 2, then 3.. Until 10 to see if it is indeed always result to 13!

8. Write a C program that will compute the total bill of a hospital patient. The total bill includes any costs incurred for the entire stay in the hospital less any insurance deductible, if any. The user inputs are as follow: (a) Number of days in hospital; (b) Surgery cost; (c) medication cost; (d) Miscellaneous cost; (e) cost per day stay in the hospital; and (f) insurance deductible amount. Your program must compute the following for insurance purposes: (i) total cost; (ii) total cost less any insurance deductible; and (iii) total cost less cost of medication and insurance deductible.

** End for Now **