ASSIGNMENT

No. 1 – Mid Term 1

Md Asaduzzaman Atik (2023-1-60-130)
2023-1-60-130@std.ewubd.edu

Table of Contents

FILE: PRACTICE 1.PDF

Basic Introductory Problems (Total 15 questions)

1. Program that will print “Hello World”.

Code:

```
#include <stdio.h>

int main() {


     printf("Hello World\n");

     return 0;

} //main
```

Output:

```
Hello World

Process returned 0 (0x0)   execution time : 0.046 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

2. Program that will use newline/tab and print the following segment.

Code:

```
#include <stdio.h>

int main() {

     printf("Hello World\n"
               "This is my first program.\tC is 
fun.\n");

     return 0;

} //main
```

Output:

```
Hello World

This is my first program.       C is fun.


Process returned 0 (0x0)   execution time : 0.040 s

Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

3. Program that will print the following segment.

The question is – “How to write a \comment/ in C programming language?”

Code:

```
#include <stdio.h>

int main() {
     printf("The question is - \"How to write a\n"
               "\\comment/ in C programming
language?\"\n");

     return 0;

} //main
```

Output:

```
The question is - "How to write a
\comment/ in C programming language?"

Process returned 0 (0x0)   execution time : 0.040 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

4. Program that will declare an integer, a floating point number, a character. Then it will initialize them with values and print those values.

Code:

```
#include <stdio.h>

int main() {

    // declaring variables
    int integer_value;
    float floating_point_value;
    char character_value;

    // initializing variables with values
```

```
     integer_value = 5;
     floating_point_value = 3.141593;
     character_value = 'a';

     // printing variables with their initialized
values
     printf("The integer value: %d\n"
               "The floating point value: %f\n"
               "The character value: %c\n",
integer_value, floating_point_value,
character_value);

     return 0;

} //main
```

Output:

```
The integer value: 5
The floating point value: 3.141593
The character value: a

Process returned 0 (0x0)   execution time : 0.040 s
Press any key to continue.
```

Observation:

Specified requirements and instructions are fulfilled

5. Program that will do the followings:
   a) Declare a variable uninitialized
   b) Declare and initialize a variable in one statement
   c) Declare and initialize multiple variables with different values in one statement

d) Declare and initialize multiple variables with the same value in one statement

Code:

```
#include <stdio.h>

int main() {

     int i;
     int j = 0;
     int k = 1, l = 2, m = 3;
     int n, o, p;
     n = o = p = 5;

     return 0;
} // main
```

Output:

```
N/A
```

Observation:

Specified requirements and instructions are fulfilled

6. Program that will take your age in year(s) as input and print it.

Code:

```
#include <stdio.h>

int main() {

     int age;
```

```
    // getting age from user input
    scanf("%d", &age);

    // printing inputted age
    printf("My age is: %d\n", age);

    return 0;

} //main
```

Output:

➔ For input: 20

```
20
My age is: 20

Process returned 0 (0x0)   execution time : 1.987 s
Press any key to continue.
```

➔ For input: 21

```
21

My age is: 21


Process returned 0 (0x0)   execution time : 1.560 s

Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

7. Program that will receive the values of an integer, a floating point number, a character from

the keyboard and print those values.

Code:

```
#include <stdio.h>

int main() {

     // declaring variables
     int integer_value;
     float floating_point_value;
     char character_value;

     // initializing variables with user inputted
data
     scanf("%d", &integer_value);
     scanf("%f", &floating_point_value);
     scanf("%*c%c", &character_value);

     // printing variables with their initialized
values
     printf("The integer value: %d\n"
               "The floating point value: %f\n"
               "The character value: %c\n",
integer_value, floating_point_value,
character_value);

     return 0;

} //main
```

Output:

➔ For input: 5

```
3.141593
A
```

```
5
3.141593
A
The integer value: 5
The floating point value: 3.141593
The character value: A

Process returned 0 (0x0)   execution time : 1.991 s
Press any key to continue.
```

➔ For input: `100 1.618 z`

```
100 1.618 z
The integer value: 100
The floating point value: 1.618000
The character value: z

Process returned 0 (0x0)   execution time : 1.324 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

8. Program that will take three integer numbers from keyboard but assign only the first and last inputs to variables and skip any assignment of the middle one.

Code:

```
#include <stdio.h>
```

```
int main() {

     int first_value, last_value;

     // initializing variables with user inputted
data
     scanf("%d%*d%d", &first_value, &last_value);

     printf("First Value = %d, Last Value = %d\n",
first_value, last_value);

     return 0;

} //main
```

Output:

- ➔ For input: 20 50 100

```
20 50 100
First Value = 20, Last Value = 100

Process returned 0 (0x0)   execution time : 0.994 s
Press any key to continue.
```

- ➔ For input: 33 75 22

```
33 75 22

First Value = 33, Last Value = 22


Process returned 0 (0x0)   execution time : 1.962 s

Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

9. Program that will declare a variable from each data type: double, boolean. Then it will initialize them with values and print them.

Code:

- Frist variant

```
#include <stdio.h>
#include <stdbool.h> // For `bool` data type
#include <math.h> // For program helper
functionality

int main() {

     double double_value;
     bool boolean_value;

     double_value = 3.1400001; /** last digit kept 1
for comparison purpose
                                        as last most
digits are 0 (zero).
                                        **/
     /** double_value = 1.6180397 **/ /** there is
no need to worry because the last

     most digit is non zero [7 (seven)]

     **/
     boolean_value = true;

     // program helper
     int count = 0;
    double decimal_part = double_value -
(int)double_value;

    while (fabs(decimal_part) > 0.000001) {
        decimal_part *= 10;
        count++;
        decimal_part -= (int)decimal_part;
    }

    if(double_value > 1000000 || count > 6) {
          printf("The double value: %.6e\n",
double_value);
    } else{
          printf("The double value: %lf\n",
double_value);
    }

     printf("The boolean value: %d\n",
boolean_value);

     return 0;

} //main
```

- Second variant

```
#include <stdio.h>
#include <stdbool.h> // For `bool` data type

int main() {

     double double_value;
     bool boolean_value;

     double_value = 3.140000;
     boolean_value = true;

     printf(//"The double value: %lf\n"
               "The double value: %e\n"
               //"The double value: %g\n"
               "The boolean value: %d\n",
               //double_value,
               double_value,
                    //double_value,
                    boolean_value);

       return 0;

} //main
```

Output:

- First variant

```
The double value: 3.140000e+000
The boolean value: 1

Process returned 0 (0x0)   execution time : 0.040 s
Press any key to continue.
```

- Second variant

```
The double value: 3.140000e+000
The boolean value: 1

Process returned 0 (0x0)   execution time : 0.038 s
Press any key to continue.
```

Observation:

Though specified requirements and instructions are fulfilled, and there is no mismatch between *Output* and *Answer*, for the double data type I think the question remains unclear according to the answer. If we look at the answer (Sample output 1 & 2), in first answer's output, the double value is processed as an exponential value (`3.140000e+00`), and in the second answer's output, the double value is processed as a regular double value (`1.618039`), here comes the real confusion.

According to my current knowledge, we can't process numbers in exponential and regular double values with the same c specifier.

If we use %e specifier for the double data type, this will process our value every time in exponential. If we use %g specifier for the double data type, this will process our value as regular value until our value exceed the digit limit 6, if the number of digits becomes 7 or higher, then our value will process as exponential, but this will only happen if the number is not a floating-point number (data type can be floating type). If we use %lf specifier for the double data type, there is no chance getting exponential output except predefined exponential value.

I have tried to solve the above problem in two ways keeping all things in mind. One is automatic exponential based on digits count in both decimal integers and digits after floating point, other is manual.

*** Seeking attention to review this code carefully

10. Program that will declare a variable from each data type: long int, long long int, long double, short int. Then it will initialize them with values and print them.

Code:

```
#include <stdio.h>
#include <float.h> // For getting the floating point system info

int main() {

    long int long_int_value;
    long long int long_long_int_value;
    long double long_double_value;
    short int short_int_value;

    long_int_value = 2147483647;
    long_long_int_value = 9223372036854775807;
    //long_double_value = 1.1E+4932;
    long_double_value = LDBL_MAX;
    short_int_value = 32767;

    printf("The long int value: %li\n"
               "The long long int value: %lli\n"
               //"The long double value: %Lf\n"
               "The long double value: %Le\n"
               "The short int value: %hd\n",
               long_int_value,
               long_long_int_value,
               //long_double_value,
               long_double_value,
               short_int_value);

     return 0;

} //main
```

Output:

```
The long int value: 2147483647
The long long int value: 9223372036854775807
The long double value: -1.#QNAN0e+000
The short int value: 32766

Process returned 0 (0x0)   execution time : 0.041 s
Press any key to continue.
```

Observation:

Specified requirements and instructions are fulfilled.

- Note: According to my thought, for the sample output 1 & 2, the long double answer seems indicating the maximum and the minimum capacity for long double data type, let's analyze just the first sample output answer long double value (1.1E+4932), though 1.1E+4932 (similar to approx. 118973149535723176502126385303097020516906332229462420044032373389173700552297072261641029033652888285354569780749557731442744315367028843419812557385374367867359320070697326320191591828296152436552951064679108661431179063216977883889613478656060039914875343321145491116008867984515486651285234014977303760000912547939396622315138362241

```
783854274391783813871780588948754057516822634765
923557697480511372564902088485522249479139937758
502601177354918009979622602685950855888360815984
690023564513234659447638493985927645628457966177
293040780660922910271504608538808795932778162298
682754783076808004015069494230341172895777710033
571401055977524212405734700738625166011082837911
962300846927720096515350020847447079244384854591
288672300061908512647211195136146752763351956292
759795725027800298079590419313960302147099703527
646744553092202267965628099149823208332964124103
850923918473478612192169721054348428704835340811
304257300221642134891734717423480071488075100206
439051723424765600472176809648610799494341570347
632064355862420744350442438056613601760883747816
538902780957697597728686007148702828795556714140
463261583262360276289631617397848425448686060994
827086796804807870251185893083854658422304090880
599629459458620190376604844679092600222541053077
590106576067134720012584640695703025713896098375
799892695455305236856075868317922311363951946885
088077187210470520395758748001314313144425494391
994017575316933939236688185618912993172910425292
123683515992232205099800167710278403536014082929
639811512287776813570604578934353545169653956125
404884644716978689321167108722908808277835051822
885764606221873970285165508372099234948333443522
898475123275372663606621390228126470623407535207
172405866507951821730346378263135339370677490195
019784169044182473806316282858685774143258116536
404021840272491339332094921949842244273042701987
304453662035026238695780468200360144729199712309
553005720614186697485284685618651483271597448120
312194675168637934309618961510733006555242148519
520176285859509105183947250286387163249416761380
499631979144187025430270675849519200883791516940
158174004671147787720145964446117520405945350476
472180797576111172084627363927960033967047003761
337450955318415007379641260504792325166135484129
188421134082301547330475406707281876350361733290
800595189632520707167390454777712968226520622565
143991937680440029238090311243791261477625596469
422198137514696707944687035800439250765945161837
```

```
981185939204954403611491531078225107269148697980
```

7747029662545110861548958395087796755464137944895960527975209874813839762578592105756284401759349324162148339565350189196811389091843795734703269406342890087805846940352453479398080674273236297887100867175802531561302356064878709259865288416350972529537091114317204887747405539054009425375424119317944175137064689643861517718849867010341532542385911089624710885385808688837777258648564145934262121086647588489260031762345960769508849149662444156604419552086811989770240.000000) is the maximum capacity of long double data type, it can’t be manually initialized to a long data typed variable, but if initialized 1.1E+4932 intentionally it will ended up resulting inf means infinity, but we can output the maximum or minimum value for long double data type with the help of float.h header file. It’s give us a bunch of constant variable, and from them LDBL_MAX provide us the maximum value or capacity of long double data type that the users system supports, and LDBL_MIN for the minimum. We can initialize these constant variable to any long double data type and print with printf() function and %Lf or %Le specifier, here %Lf specifier will output all the digits, but %Le will give exponential output.

In my provided code I tried to output my systems maximum value or capacity with %Le specifier, but in my output I got -1.#QNAN0e+000, that is a representation of a “quiet NaN” (Not a Number) value in scientific notation. A quiet NaN propagates through most arithmetic operations without raising an exception. I’m getting this value because my system doesn’t support the long double data type. If my system supports the long double data type I could possibly get the output quite similar to 1.18973e+4932.

*** Seeking attention to review this code carefully

11. Program that will declare a variable from each data type: unsigned int, unsigned long int, unsigned long long int, unsigned short int. Then it will initialize them with values and print them.

Code:

```
#include <stdio.h>
```

```
int main() {

     unsigned int unsigned_int_value;
     unsigned long int unsigned_long_int_value;
     unsigned long long int
unsigned_long_long_int_value;
     unsigned short int unsigned_short_int_value;

     unsigned_int_value = -4294967296;
     unsigned_long_int_value = -4294967296;
     unsigned_long_long_int_value = -
18446744073709551616;
     unsigned_short_int_value = -65536;

     printf("The unsigned int value: %u\n"
               "The unsigned long int value: %lu\n"
               "The unsigned long long int value:
%llu\n"
               "The unsigned short int value:
%hu\n",
               unsigned_int_value,
               unsigned_long_int_value,
               unsigned_long_long_int_value,
               unsigned_short_int_value);

     return 0;

} //main
```

Output:

```
The unsigned int value: 0
The unsigned long int value: 0
The unsigned long long int value: 0
The unsigned short int value: 0

Process returned 0 (0x0)   execution time : 0.042 s
Press any key to continue.
```

Observation:

Specified requirements and instructions are fulfilled

12. Program that will define a constant using “CONST” and print the value.

Code:

```
#include <stdio.h>

int main() {

    const float PI = 3.14;

    printf("The value of pi: %.2f\n", PI);

    return 0;
} // main
```

Output:

```
The value of pi: 3.14

Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

Observation:

Specified requirements and instructions are fulfilled

13. Program that will define a constant using “DEFINE” and print the value.

Code:

```
#include <stdio.h>
#define PI 3.14
```

```
int main() {

    printf("The value of PI: %.2f\n", PI);

    return 0;

} //main
```

Output:

```
The value of PI: 3.14

Process returned 0 (0x0)   execution time : 0.045 s
Press any key to continue.
```

Observation:

Specified requirements and instructions are fulfilled

14. Program that will define a global and a local variable with the same name but with different values, and then do the following steps in order-
    A. Print the value of the variable before defining the local variable
    B. Print the value of the variable after defining the local variable
    C. Explicitly print the value of the variable as global

Code:

```
#include <stdio.h>

int x = 10;

int main() {

    printf("A. Global: %d\n", x);

    int x = 20;
    printf("B. Local: %d\n", x);

    {
        extern int x;
        printf("C. Global: %d\n", x);
    }

    return 0;

} //main
```

Output:

```
A. Global: 10

B. Local: 20

C. Global: 10


Process returned 0 (0x0)   execution time : 0.042 s

Press any key to continue.
```

Observation:

Specified requirements and instructions are fulfilled

15. Program that will take an floating point number as input from the keyboard and use printf function to perform the followings:

    (a) Print the number right justified within 10 columns
    (b) Print the number to be right justified to 2 columns (Assuming the input has more
    than 2 digits)
    (c) Print the number rounded to two decimal places
    (d) Print the number rounded to integer (without using conversion or type casting)
    (e) Prints the number in exponential notation/scientific notation

Code:

```
#include <stdio.h>

int main() {

    float num;

    scanf("%f", &num);

    printf("(a) Val:%10f\n"
                "(b) Val:%2f\n"
                "(c) Val:%.2f\n"
                "(d) Val:%.0f\n"
                "(e) Val: %e\n", num, num, num, num,
num);

    return 0;

} //main
```

Output:

➔ For input: 123.098

```
123.098
(a) Val:123.098000
(b) Val:123.098000
(c) Val:123.10
(d) Val:123
(e) Val: 1.230980e+002

Process returned 0 (0x0)   execution time : 9.944 s
Press any key to continue.
```

Observation:

According to my observation, I think there is a visual mismatch for instruction (a), in answer for sample input

```
123.098
```

the sample output for the instruction (a) is

```
(a) Val:       123.098000
```

And the output for my code is

```
(a) Val:123.098000
```

But theoretically, I think I fulfilled specified requirements and instructions,

*** Seeking attention to review this code carefully

FILE: PRACTICE 2.PDF

Operator Related Problems (Total 15 questions)

1. Program that will take two numbers X and Y as inputs, then calculate and print the values of their addition, subtraction, multiplication, division (quotient and reminder).

Code:

```
#include <stdio.h>

int main() {

    double x, y;

    scanf("%lf %lf", &x, &y);

    printf("Addition: %g\n"
               "Subtraction: %g\n"
               "Multiplication: %g\n"
               "Quotient : %d\n"
               "Reminder: %d\n", x+y, x-y, x*y,
(int)x/(int)y, (int)x%(int)y);

    return 0;

} // main
```

Output:

➔ For input: 5 10

```
5 10
Addition: 15
Subtraction: -5
Multiplication: 50
Quotient : 0
Reminder: 5

Process returned 0 (0x0)   execution time : 1.453 s
Press any key to continue.
```

➔ For input: `-5 10.5`

```
-5 10.5
Addition: 5.5
Subtraction: -15.5
Multiplication: -52.5
Quotient : 0
Reminder: -5

Process returned 0 (0x0)   execution time : 1.125 s
Press any key to continue.
```

Observation:

There is no mismatch between *Output* and *Answer*, except for the second answer’s output’s reminder result. In answer the reminder is `-48`, but from my code the output is `-5`, and I think there is no possible way to get `-48` for the input `-5 10.5`. For clearance, I tried the calculation in other languages (code added below) and compare those results with mine and the answer, from comparison I think my calculation and code contains no error, there is error in answer.

➢ In PHP

```
asada@KB-PC-01-WIN10-1 MINGW64 ~ (main)
$ php -a
Interactive shell

php > echo -5%10.5;
PHP Deprecated:  Implicit conversion from
float 10.5 to int loses precision in php shell
code on line 1

Deprecated: Implicit conversion from float
10.5 to int loses precision in php shell code
on line 1
-5
php >
```

➢ In NODE/JS:

```
asada@KB-PC-01-WIN10-1 MINGW64 ~ (main)
$ node
Welcome to Node.js v18.14.2.
Type ".help" for more information.
> -5%10.5
-5
>
```

- In PYTHON:

```
asada@KB-PC-01-WIN10-1 MINGW64 ~ (main)
$ py
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7
2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or
"license" for more information.
>>> -5%10.5
5.5
>>>
```

** Different output, because Python does not automatically convert the floating-point number to integer, let's try manually:

```
asada@KB-PC-01-WIN10-1 MINGW64 ~ (main)
$ py
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7
2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or
"license" for more information.
>>> -5%10
5
>>>
```

** Again, different output, in other languages, including my code the output for the remainder is −5 (negative), but in python the output for the remainder is 5 (positive).

In C, PHP, and NODE/JS, the modulo operation on a negative number results in a negative remainder. So, −5%10.5 (automatic conversion to −5%10) would result in −5, since −5 is the remainder when −5 is divided by 10.5.

In Python, the modulo operation is performed differently. The sign of the result is determined by the divisor (10.5 [manual conversion to 10] in this case), not the dividend (−5). Since 10 is positive, the result of −5%10 is positive, and the remainder is calculated accordingly, resulting in 5.

Whether it's a negative number or a positive number the ultimate result of −5%10.5 is |5|, there is no chance to be −48, I think.

*** Seeking attention to review this code carefully

2. Program that will calculate the circumference of a circle having radius r.
   Area, A = 2 * Pi * r

Code:

```c
#include <stdio.h>

int main() {

	const double PI = 3.1415926535897;
	double r;

	scanf("%lf", &r);

	printf("Area: %0.2lf", 2*PI*r);

	return 0;

} //main
```

Output:

➔ For input: 5

```
5
Area: 31.42
Process returned 0 (0x0)   execution time : 7.342 s
Press any key to continue.
```

➔ For input: 10.5

```
10.5
Area: 65.97
Process returned 0 (0x0)   execution time : 2.495 s
Press any key to continue.
```

Observation:

There is a mismatch between *output* and *answer* because of the difference in the value of PI, moreover the rest of the instructions are fulfilled.

3. Program that will take two numbers (a, b) as inputs and compute the value of the equation
– (Without using math.h)

X = (3.31 * a2 + 2.01 * b3) / (7.16 * b2 + 2.01 * a3)

Code:

```
#include <stdio.h>

int main() {

     // defining variable with their perspective
data type
     double a, b, x;

     // getting user input and storing to `a` and
`b`
     scanf("%lf %lf", &a, &b);
```

```
    // implementing equation and calculation and
storing the result into `x`
    x = (3.31*(a*a)+2.01*(b*b*b)) /
(7.16*(b*b)+2.01*(a*a*a));

    // printing the value of `x`
    printf("X = %lf", x);

    // returning void
    return 0;

} //main
```

Output:

➔ For input: 5 10.5

```
5 10.5
X = 2.315475
Process returned 0 (0x0)   execution time : 2.574 s
Press any key to continue.
```

➔ For input: 100 -250

```
100 -250
X = -12.766287
Process returned 0 (0x0)   execution time : 4.494 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

4. Program that will increment and decrement a number X by 1 inside the printf function.
(Use ++ and - - operators)

Code:

```c
#include <stdio.h>

int main() {

    int x;

    scanf("%d", &x);

    const int helper = x;

    x = helper;
    printf("X++ : %d\n", x++);

    x = helper;
    printf("++X : %d\n", ++x);

    x = helper;
    printf("X-- : %d\n", x--);

    x = helper;
    printf("--X : %d\n", --x);

    return 0;

} //main
```

Output:

➔ For input: 5

```
5
X++ : 5
++X : 6
X-- : 5
--X : 4

Process returned 0 (0x0)   execution time : 1.116 s
Press any key to continue.
```

➔ For input: -5

```
-5
X++ : -5
++X : -4
X-- : -5
--X : -6

Process returned 0 (0x0)   execution time : 2.161 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

5. Program that will increment and decrement a number X by Y. (Use += and -= operators)

Code:

```c
#include <stdio.h>

int main() {

    int x, y;

    scanf("%d %d", &x, &y);

    const int helper = x;

    x += y;
    printf("Incremented Value: %d\n", x);

    x = helper;
    x -= y;
    printf("Decremented Value: %d\n", x);

    return 0;

} //main
```

Output:

➔ For input: 5 10

```
5 10
Incremented Value: 15
Decremented Value: -5

Process returned 0 (0x0)   execution time : 1.998 s
Press any key to continue.
```

➔ For input: -5 5

```
-5 5
Incremented Value: 0
Decremented Value: -10

Process returned 0 (0x0)   execution time : 2.583 s
Press any key to continue.
```

Observation:

Though there is no mismatch between *Output* and *Answer* for sample 2, but in sample 1, for the 5 10 input the incremented value is 10 that is impossible in every way. Because x += y; means, x = x + y; so for the input 5 10, it's ultimately x = 5 + 10; which results 15. I think the answer for sample 1 is incorrect.

*** Seeking attention to review this code carefully

6. Program that will multiply and divide a number X by Y. (Use *= and /= operators)

Code:

```
#include <stdio.h>

int main() {

    int x, y;
```

```
        scanf("%d %d", &x, &y);

        const int helper = x;

        x *= y;
        printf("Multiplication: %d\n", x);

        x = helper;
        x /= y;
        printf("Division: %d\n", x);

        return 0;

    } //main
```

Output:

➔ For input: 56 10

```
56 10
Multiplication: 560
Division: 5

Process returned 0 (0x0)   execution time : 1.717 s
Press any key to continue.
```

➔ For input: -56 -10

```
-56 -10
Multiplication: 560
Division: 5

Process returned 0 (0x0)   execution time : 4.940 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

7. Program that will declare and initialize an integer and a floating point number. Then it will perform floating to integer and integer to floating conversions using

(a) Assignment operation
(b) Type casting

Code:

```
#include <stdio.h>

int main() {

    int integer_value = -150;
    float floating_point_value = 123.125;

    scanf("%d %f", &integer_value,
&floating_point_value);

    int float_to_int_assignment =
floating_point_value;
    float int_to_float_assignment = integer_value;

    printf("Assignment: %f assigned to an int
produces %d\n", floating_point_value,
float_to_int_assignment);
    printf("Assignment: %d assigned to a float
produces %f\n", integer_value,
int_to_float_assignment);
    printf("Type Casting: (float) %d produces %f\n",
integer_value, (float)integer_value);
    printf("Type Casting: (int) %g produces %d\n",
floating_point_value, (int)floating_point_value);

    return 0;

} //main
```

Output:

➔ For input: `-150 123.125`

```
-150 123.125
```

```
Assignment: 123.125000 assigned to an int produces
123
Assignment: -150 assigned to a float produces -
150.000000
Type Casting: (float) -150 produces -150.000000
Type Casting: (int) 123.125 produces 123

Process returned 0 (0x0)   execution time : 3.364 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

8. Program that will take two numbers as inputs and print the maximum value. (Using conditional operator - ?)

Code:

```
#include <stdio.h>

int main() {

    int x, y, max;

    scanf("%d %d", &x, &y);

    printf("Max: %d", x > y ? x : y);

    return 0;

} //main
```

Output:

➔ For input: 20 100

```
20 100
Max: 100
```

```
Process returned 0 (0x0)   execution time : 2.242 s
Press any key to continue.
```

➔ For input: `50 -20`

```
50 -20
Max: 50

Process returned 0 (0x0)   execution time : 3.226 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

9. Program that will evaluate the following equations -

$X = a - b / 3 + c * 2 - 1$

$Y = a - (b / (3 + c) * 2) - 1$

$Z = a - ((b / 3) + c * 2) - 1$

Code:

```
#include <stdio.h>

int main() {

    double a, b, c, x, y, z;

    scanf("%lf %lf %lf", &a, &b, &c);

    x = a - b / 3 + c * 2 - 1;
    y = a - ( b / ( 3 + c ) * 2 ) - 1;
    z = a - ( ( b / 3 ) + c * 2 ) - 1;

    printf("X = %g\n"
           "Y = %g\n"
           "Z = %g\n", x, y, z);

    return 0;

} //main
```

Output:

➔ For input: 9 12 3

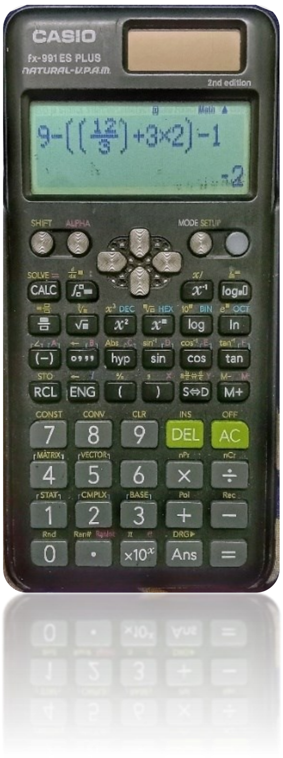
```
9 12 3
X = 10
Y = 4
Z = -2

Process returned 0 (0x0)   execution time : 2.762 s
Press any key to continue.
```

Observation:

Though the calculation and code done by the exact instructions, there is a mismatch for Z output for the input 9 12 3. Output of my code give −2 for Z, but in answer, output for Z is −1. For clearance, I calculated the Z equation with physical scientific calculator **CSIO** fx-991ES PLUS, in calculator the output of Z is also −2 (picture added below). So, my code and output are right, and the answer is incorrect, I think.

- Attachment – Calculation of Z equation with **CASIO** fx-991ES PLUS



*** Seeking attention to review this code carefully

10. Program that will take a, b & c as inputs and decide if the statements are True (1) of False (0)

    a) (a + b) ≤ 80

    b) ! (a + c)

    c) a! = 0

Code:

```
#include <stdio.h>
#include <stdbool.h>

int main() {

     int a, b, c;

     scanf("%d %d %d", &a, &b, &c);

     printf("a) %d\n"
               "b) %d\n"
               "c) %d\n",
               (a+b) <= 80 ? true : false,
               !(a+c) ? true : false,
               a != 0 ? true : false);

     return 0;

} //main
```

Output:

➔ For input: `10 -10 0`

```
10 -10 0
a) 1
b) 0
c) 1

Process returned 0 (0x0)   execution time : 1.117 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

11. Program that will take a, b & c as inputs and decide if the statements are True (1) of False (0)

    1) (a + b) ≤ 80 && b ≥ 0
    2) (a − b) == 0 | |c! = 0
    3) a! = b | |(b < a)&&c > 0

Code:

```
#include <stdio.h>
#include <stdbool.h>

int main() {

     int a, b, c;

     scanf("%d %d %d", &a, &b, &c);

     printf("1) %d\n"
               "2) %d\n"
               "3) %d\n",
               (a + b) <= 80 && b >= 0 ? true :
false,
               (a - b) == 0 || c != 0 ? true :
false,
               a != b || (b < a) && c > 0 ? true :
false);

     return 0;

} //main
```

Output:

➔ For input: `10 -10 0`

```
10 -10 0
```

```
1) 0
2) 0
3) 1

Process returned 0 (0x0)   execution time : 4.365 s
Press any key to continue.
```

Observation:

There is a mismatch between *Output* and *Answer* in second expression, my code returns `0`, but answer given here is `1`.

Let's deep dive into the second expression with the inputs `10 -10 0`:

2) `(a - b) == 0 || c != 0`:
- Substituting the values of a, b, and c, we get `(10 - (-10)) == 0 || 0 != 0`
- Simplifying, we get `20 == 0 || 0 != 0`
- The first condition is false `(20 == 0)`, and the second condition is also false `(0 != 0)`
- Therefore, the overall result of this expression should have been false (`0`)
- The output line for this expression should have been false `2) 0`

I have no explanation for this mismatch.

*** Seeking attention to review this code carefully

12. Program that will take calculate the roots of a quadratic equation (a.x2 + b.x + c = 0) from the formula, (here, dot (.) stands for multiplication) –

$$root = \frac{-b \pm \sqrt{b^2 - 4.a.c}}{2.a}$$

Code:

```
#include <stdio.h>
#include <math.h>

int main() {

    float a, b, c, discriminant, root1, root2;
    scanf("%f %f %f", &a, &b, &c);

    discriminant = b*b - 4*a*c;

    if (discriminant > 0) {
        root1 = (-b + sqrt(discriminant)) / (2*a);
        root2 = (-b - sqrt(discriminant)) / (2*a);
        printf("%.2f %.2f", root1, root2);
    }else if (discriminant == 0) {
        root1 = root2 = -b / (2*a);
        printf("%.2f %.2f", root1, root2);
    }else {
        printf("Imaginary");
    }

    return 0;

} //main
```

Output:

➔ For input: 2 4 -16

```
2 4 -16
2.00 -4.00
Process returned 0 (0x0)   execution time : 4.045 s
Press any key to continue.
```

➔ For input: 1 2 3

```
1 2 3
Imaginary
Process returned 0 (0x0)   execution time : 3.072 s
Press any key to continue.
```

Observation:

There is no mismatch found between *Output* and *Answer*

I took help from the internet about how to solve quadratic equation and for what condition it is imaginary (https://byjus.com/jee/quadratic-equations/).

13. Program that will evaluate the equation $2\cos^2 x - \sqrt{3}\sin x + \sin\frac{x}{2}$; where 1 <= x <= 180 [No checking needed]

Code:

```
#include <stdio.h>
#include <math.h>

int main() {

    double x, result;

    scanf("%lf", &x);

    x = x * M_PI / 180.0;

    result = 2 * pow(cos(x), 2) - sqrt(3) * sin(x) +
sin(x/2);

    printf("%lf\n", result);

    return 0;

} //main
```

Output:

- ➔ For input: 30

```
30
0.892794

Process returned 0 (0x0)   execution time : 2.092 s
Press any key to continue.
```

- ➔ For input: 120

```
120
-0.133975

Process returned 0 (0x0)   execution time : 1.608 s
```

```
Press any key to continue.
```

➔ For input: 180

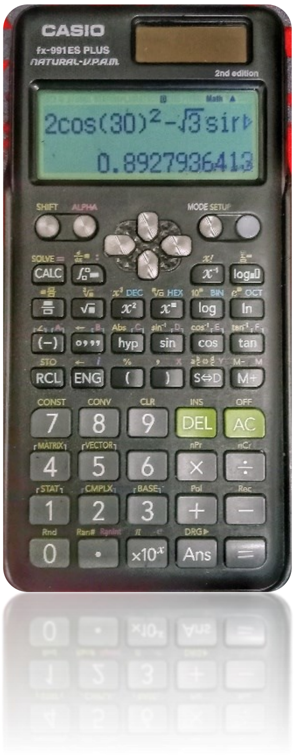
```
180
3.000000

Process returned 0 (0x0)   execution time : 2.549 s
Press any key to continue.
```

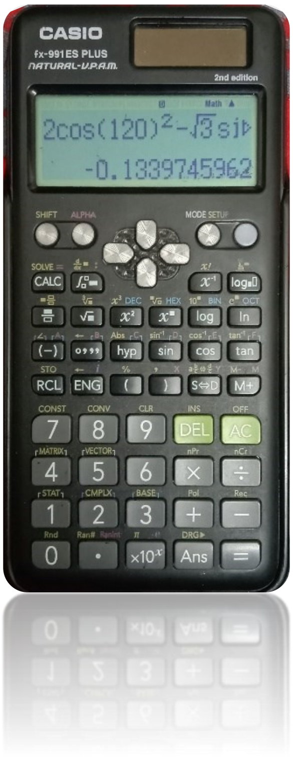
Observation:

No match found between *Output* and *Answer*. Why and how, I have no explanation. For clearance, I calculated the expression with physical scientific calculator CASIO fx-991ES PLUS with every sample input (picture added below), there is also mismatch with the calculator and answer, but the output from the calculator matches my code output.

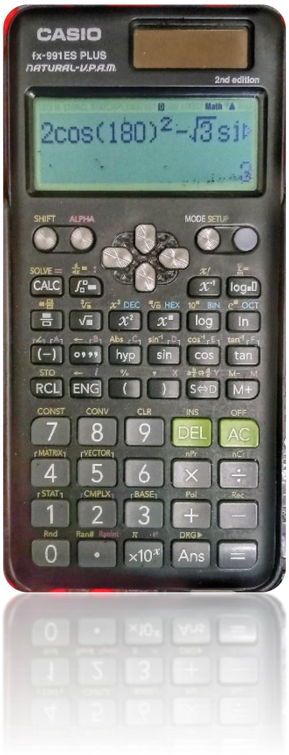
- Attachment – Calculation for the expression with CASIO fx-991ES PLUS
    - For Input: 30



- For input: 120



- For input: 180



*** Seeking attention to review this code carefully

14. Program that will take a floating point number X as input and evaluate A,B,C where-

A = Value when X is rounded up to the nearest integer

B = Value when X is rounded down to the nearest integer

C = Absolute value of X

Code:

```
#include <stdio.h>
#include <math.h>

int main() {

    float x;

    scanf("%f", &x);

    printf("A = %g, B = %g, C = %g\n", ceil(x),
floor(x), fabs(x));

    return 0;

} // main
```

Output:

➔ For input: 10.6

```
10.6
A = 11, B = 10, C = 10.6

Process returned 0 (0x0)   execution time : 2.484 s
Press any key to continue.
```

➔ For input: -77.9

```
-77.9
A = -77, B = -78, C = 77.9

Process returned 0 (0x0)   execution time : 3.956 s
Press any key to continue.
```

Observation:

No mismatch found for the input 10.6, but for the input -77.9, there is a big mismatch found. In general sense the nearest rounded up value for -77.9 is -77 and rounded down value is -78, but in answer for input -77.9 the rounded-up value is 78, and rounded down value is 77 (all are positive output). For clearance, I tried rounding up and down the input -77.9 in other languages (code added below), output from those tries matches with my output.

- In PHP:

```
asada@KB-PC-01-WIN10-1 MINGW64 ~ (main)
$ php -a
Interactive shell

php > echo ceil(-77.9);
-77
php > echo floor(-77.9);
-78
php >
```

- In NODE/JS:

```
asada@KB-PC-01-WIN10-1 MINGW64 ~ (main)
$ node
Welcome to Node.js v18.14.2.
Type ".help" for more information.
> Math.ceil(-77.9);
-77
> Math.floor(-77.9);
-78
>
```

- In Pyhton:

```
asada@KB-PC-01-WIN10-1 MINGW64 ~ (main)
$ py
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7
2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or
"license" for more information.
```

```
>>> import math
>>> math.ceil(-77.9)
-77
>>> math.floor(-77.9)
-78
>>>
```

*** Seeking attention to review this code carefully

15. Program to find size of int, float, double and char of the system.

Code:

```
#include <stdio.h>

int main() {

     printf("Size of int in byte(s) = %d\n"
               "Size of float in byte(s) = %d\n"
               "Size of double in byte(s) = %d\n"
               "Size of char in byte(s) = %d\n",
sizeof(int), sizeof(float), sizeof(double),
sizeof(char));

     return 0;

} //main
```

Output:

```
Size of int in byte(s) = 4
Size of float in byte(s) = 4
Size of double in byte(s) = 8
Size of char in byte(s) = 1

Process returned 0 (0x0)   execution time : 0.040 s
Press any key to continue.
```

Observation:

Specified requirements and instructions fulfilled.

FILE: 03. CONDITION_RELATED_PROBLEMS.PDF

Operator Related Problems (Total 15 questions)

1. Program that will decide whether a number is positive or not.

Code:

```
#include <stdio.h>

int main() {
	double x;

	scanf("%lf", &x);

	if(x >= 0) {
		printf("Positive\n");
	} else {
		printf("Negative\n");
	}

	return 0;

} //main
```

Output:

➔ For input: 100

```
100
Positive

Process returned 0 (0x0)   execution time : 1.544 s
Press any key to continue.
```

➔ For input: -11.11

```
-11.11
Negative

Process returned 0 (0x0)   execution time : 3.016 s
Press any key to continue.
```

➔ For input: 0

```
0
Positive

Process returned 0 (0x0)   execution time : 0.782 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

2. Program that will decide whether a number is even or odd.

Code:

```
#include <stdio.h>

int main() {
     int x;

     scanf("%d", &x);

     if(x%2 == 0) {
          printf("Even\n");
     } else {
          printf("Odd\n");
     }

     return 0;

} //main
```

Output:

➔ For input: 50

```
50
Even
```

```
Process returned 0 (0x0)   execution time : 1.237 s
Press any key to continue.
```

➔ For input: -77

```
-77
Odd

Process returned 0 (0x0)   execution time : 1.779 s
Press any key to continue.
```

➔ For input: 0

```
0
Even

Process returned 0 (0x0)   execution time : 1.284 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

3. Program that will take an integer of length one from the terminal and then display the digit in English.

Code:

➔ First variant:

```
#include <stdio.h>

int main() {

	int x;

	scanf("%1d", &x);

	if(x == 0) {
		printf("zero");
	} else if(x == 1) {
		printf("one");
```

```
    } else if(x == 2) {
        printf("two");
    } else if(x == 3) {
        printf("three");
    } else if(x == 4) {
        printf("four");
    } else if(x == 5) {
        printf("five");
    } else if(x == 6) {
        printf("six");
    } else if(x == 7) {
        printf("seven");
    } else if(x == 8) {
        printf("eight");
    } else {
        printf("nine");
    }

    return 0;

} //main
```

➔ Second variant:

```
#include <stdio.h>

int main() {

    int x;
    char* eng_digit[10] = {"zero", "one", "two",
"three", "four", "five", "six", "seven", "eight",
"nine"};

    scanf("%1d", &x);

    printf("%s", eng_digit[x]);

    return 0;

} //main
```

Output:

** Same output for both variant

➔ For input: 9

```
9
nine
Process returned 0 (0x0)   execution time : 1.589 s
Press any key to continue.
```

➔ For input: 0

```
0
zero
Process returned 0 (0x0)   execution time : 0.485 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

4. Program that will check whether a triangle is valid or not, when the three angles (angle value should be such that, 0 < value < 180) of the triangle are entered through the keyboard.
[Hint: A triangle is valid if the sum of all the three angles is equal to 180 degrees.]

Code:

```
#include <stdio.h>

int main() {

    double a, b, c;

    scanf("%lf %lf %lf", &a, &b, &c);

    if((a > 0 && a < 180) && (b > 0 && b < 180) && (c > 0 && c < 180)) {
        if((a + b + c) == 180) {
            printf("Yes");
```

```c
            } else {
                printf("No");
            }
        } else {
            printf("No");
        }

        return 0;

} //main
```

Output:

➔ For input: `90 45 45`

```
90 45 45
Yes
Process returned 0 (0x0)   execution time : 2.441 s
Press any key to continue.
```

➔ For input: `30 110 40`

```
30 110 40
Yes
Process returned 0 (0x0)   execution time : 2.466 s
Press any key to continue.
```

➔ For input: `160 20 30`

```
160 20 30
No
Process returned 0 (0x0)   execution time : 2.731 s
Press any key to continue.
```

➔ For input: `0 180 0`

```
0 180 0
No
Process returned 0 (0x0)   execution time : 2.428 s
Press any key to continue.
```

Observation:

5. Program that will read from the console a random positive nonzero number and determine if it is a power of 2.

Code:

```c
#include <stdio.h>

int main() {

    unsigned int num, check = 1;

    scanf("%u", &num);

    while (check < num) {
        check *= 2;
    }

    if (check == num) {
        printf("Yes\n");
    } else {
        printf("No\n");
    }

    return 0;

} // main
```

Output:

➔ For input: 1

```
1
Yes

Process returned 0 (0x0)   execution time : 1.121 s
Press any key to continue.
```

➔ For input: 512

```
512
Yes

Process returned 0 (0x0)   execution time : 1.558 s
Press any key to continue.
```

➔ For input: 1022

```
1022
No

Process returned 0 (0x0)   execution time : 1.499 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

6. Program that will read from the console a random number and check if it is a nonzero positive number. If the check is yes, it will determine if the number is a power of 2. If the check fails the program will check for two more cases. If the number is zero, the program will print “Zero is not a valid input”. Else it will print “Negative input is not valid”.

Code:

```
#include <stdio.h>

int main() {

    int num, check = 1;

    scanf("%d", &num);

    if (num > 0) {
          while (check < num) {
               check *= 2;
          }

          if (check == num) {
               printf("Yes\n");
          } else {
               printf("No\n");
          }
    } else if (num == 0) {
          printf("Zero is not a valid input\n");
    } else {
          printf("Negative input is not valid\n");
    }

    return 0;

} // main
```

Output:

- ➔ For input: 0

```
0
Zero is not a valid input

Process returned 0 (0x0)   execution time : 0.565 s
Press any key to continue.
```

- ➔ For input: 1

```
1
Yes

Process returned 0 (0x0)   execution time : 1.121 s
Press any key to continue.
```

- ➔ For input: 512

```
512
Yes

Process returned 0 (0x0)   execution time : 1.558 s
Press any key to continue.
```

- ➔ For input: 1022

```
1022
```

```
No

Process returned 0 (0x0)   execution time : 1.499 s
Press any key to continue.
```

➔ For input: -512

```
-512
Negative input is not valid

Process returned 0 (0x0)   execution time : 0.882 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

7. Program that will take two numbers X & Y as inputs and decide whether X is greater than/less than/equal to Y.

Code:

```
#include <stdio.h>

int main() {

    double x, y;

    scanf("%lf %lf", &x, &y);

    if (x > y) {
        printf("%g is grater than %g", x, y);
    } else if (x < y) {
        printf("%g is less than %g", x, y);
    } else {
        printf("%g is equal to %g", x, y);
    }

    return 0;

} //main
```

Output:

➔ For input: 5 -10

```
5 -10
5 is grater than -10
Process returned 0 (0x0)   execution time : 2.728 s
Press any key to continue.
```

➔ For input: 5 10

```
5 10
5 is less than 10
Process returned 0 (0x0)   execution time : 1.740 s
Press any key to continue.
```

➔ For input: 5 5

```
5 5
5 is equal to 5
Process returned 0 (0x0)   execution time : 1.563 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

8. Program that will decide whether a year is leap year or not.
Yes, if (Year % 4 == 0 && year % 100 != 0) || (Year % 400 ==0)

Code:

```
#include <stdio.h>

int main() {

    int year;

    scanf("%d", &year);

    if ((year%4 == 0 && year%100 != 0) || (year%400
== 0)) {
             printf("Yes\n");
    } else {
             printf("No\n");
    }

    return 0;

} //main
```

Output:

➔ For input: 2000

```
2000
Yes

Process returned 0 (0x0)   execution time : 1.464 s
Press any key to continue.
```

➔ For input: 2004

```
2004
Yes

Process returned 0 (0x0)   execution time : 1.583 s
Press any key to continue.
```

➔ For input: 2014

```
2014
No

Process returned 0 (0x0)   execution time : 2.222 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

9. Program that will categorize a single character that is entered at the terminal, whether it is an alphabet, a digit or a special character.

(Restriction: Without math.h)

Code:

```
#include <stdio.h>

int main() {

    char character;

    scanf("%c", &character);

    if ((character >= 'a' && character <= 'z') ||
(character >= 'A' && character <= 'Z')) {
        printf("Alphabet\n");
    }
    else if (character >= '0' && character <= '9') {
        printf("Digit\n");
    }
    else {
        printf("Special\n");
    }

    return 0;

} //main
```

Output:

➔ For input: z

```
z
Alphabet

Process returned 0 (0x0)   execution time : 0.879 s
Press any key to continue.
```

➔ For input: A

```
A
Alphabet
```

```
Process returned 0 (0x0)   execution time : 0.792 s
Press any key to continue.
```

➔ For input: 8

```
8
Digit

Process returned 0 (0x0)   execution time : 1.323 s
Press any key to continue.
```

➔ For input: *

```
*
Special

Process returned 0 (0x0)   execution time : 1.029 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

10. Program that will evaluate simple expressions of the form - <number1> <operator> <number2> ; where operators are (+, - , *, /). And if the operator is “/”, then check if <number2> nonzero or not.

Code:

```
#include <stdio.h>

int main() {

     double number1, number2;
     char _operator;

     scanf("%lf %c %lf", &number1, &_operator,
&number2);

     switch(_operator) {
          case '+':
               printf("Addition: %g\n", number1 +
number2);
               break;
          case '-':
               printf("Subtraction: %g\n", number1 -
number2);
               break;
          case '*':
               printf("Multiplication: %g\n",
number1 * number2);
               break;
          case '/':
               if (number2 == 0) {
                    printf("Division: Zero as
divisor is not valid!\n");
               } else {
                    printf("Division: %lf\n",
number1 / number2);
               }
               break;
          default:
               printf("No operator match!");
     }

     return 0;

} //main
```

Output:

➔ For input: `100 * 55.5`

```
100 * 55.5
Multiplication: 5550

Process returned 0 (0x0)   execution time : 3.969 s
Press any key to continue.
```

➔ For input: `100 / -5.5`

```
100 / -5.5
```

```
Division: -18.181818

Process returned 0 (0x0)   execution time : 5.138 s
Press any key to continue.
```

➔ For input: `100 / 0`

```
100 / 0
Division: Zero as divisor is not valid!

Process returned 0 (0x0)   execution time : 2.443 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

11. Program that will take the final score of a student in a particular subject as input and find his/her grade.

Code:

```c
#include <stdio.h>

int main() {

     double mark;
     char* grade;

     scanf("%lf", &mark);

     if (mark >= 90 && mark <= 100) {
          grade = "A";
     } else if (mark >= 86) {
          grade = "A-";
     } else if (mark >= 82) {
          grade = "B+";
     } else if (mark >= 78) {
          grade = "B";
     } else if (mark >= 74) {
          grade = "B-";
```

```c
    } else if (mark >= 70) {
        grade = "C+";
    } else if (mark >= 66) {
        grade = "C";
    } else if (mark >= 62) {
        grade = "C-";
    } else if (mark >= 58) {
        grade = "D+";
    } else if (mark >= 55) {
        grade = "D";
    } else if (mark >= 0 && mark < 55) {
        grade = "F";
    } else {
        grade = "Invalid";
    }

    printf("Grade: %s", grade);

    return 0;

} //main
```

Output:

➔ For input: 91.5

```
91.5
Grade: A
Process returned 0 (0x0)   execution time : 2.678 s
Press any key to continue.
```

➔ For input: 50

```
50
Grade: F
Process returned 0 (0x0)   execution time : 1.542 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

12. Program that will construct a menu for performing arithmetic operations. The user will give two real numbers (a, b) on which the arithmetic operations will be performed and an integer number (1 <= Choice <= 4) as a choice. Choice-1, 2, 3, 4 are for performing addition, subtraction, multiplication, division (quotient) respectively.

Code:

```
#include <stdio.h>

int main() {

    double a, b;
    int o;

    scanf("%lf %lf", &a, &b);
    scanf("%d", &o);

    switch(o) {
        case 1:
            printf("Addition: %g\n", a + b);
            break;
        case 2:
            printf("Subtraction: %g\n", a - b);
            break;
        case 3:
            printf("Multiplication: %g\n", a *
b);
            break;
        case 4:
            printf("Quotient: %d\n",
(int)a/(int)b);
            break;
        default:
            printf("Wrong choice\n");
    }

    return 0;

} //main
```

Output:

➔ For input: `5 10`

`3`

```
5 10
3
Multiplication: 50

Process returned 0 (0x0)   execution time : 3.334 s
Press any key to continue.
```

➔ For input: `-5 10.5`

`4`

```
-5 10.5
4
Quotient: 0

Process returned 0 (0x0)   execution time : 10.274
s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

13. Program that will construct a menu for performing arithmetic operations. The user will give two real numbers (a, b) on which the arithmetic operations will be performed and an integer number (1 <= Choice <= 4) as a choice. Choice-1, 2, 3, 4 are for performing addition, subtraction, multiplication, division respectively. If Choice-4 is selected, again the program will ask for another choice (1 <= Case <=2), where Case-1, 2 evaluate quotient and reminder respectively.

Code:

```c
#include <stdio.h>

int main() {
    double a, b;
    int o, q_or_r;

    scanf("%lf %lf", &a, &b);
    scanf("%d", &o);

    switch(o) {
        case 1:
            printf("Addition: %g\n", a + b);
            break;
        case 2:
            printf("Subtraction: %g\n", a - b);
            break;
        case 3:
            printf("Multiplication: %g\n", a *
b);
            break;
        case 4:
            scanf("%d", &q_or_r);
            switch(q_or_r) {
                case 1:
                    printf("Quotient: %d\n",
(int)a/(int)b);
                    break;
                case 2:
                    printf("Reminder: %d\n",
(int)a%(int)b);
                    break;
                default:
                    printf("Wrong choice\n");
            }
            break;
        default:
            printf("Wrong choice\n");
    }

    return 0;

} //main
```

Output:

➔ For input: `5 10`
`3`

```
5 10
3
Multiplication: 50

Process returned 0 (0x0)   execution time : 4.434 s
Press any key to continue.
```

➔ For input: `-5 10.5`
`4`
`1`

```
-5 10.5
4
1
Quotient: 0

Process returned 0 (0x0)   execution time : 9.005 s
Press any key to continue.
```

➔ For input: `-5 10.5`
`4`
`2`

```
-5 10.5
4
2
Reminder: -5

Process returned 0 (0x0)   execution time : 6.608 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer* except for sample 3. In answer the reminder is `-48`, but from my code the output is `-5`, and I think there is no possible way to get `-48` for the input `-5 10.5`. For clearance, I tried the calculation in other languages (code added below) and compare those results with mine and the answer, from comparison I think my calculation and code contains no error, there is error in answer.

- In PHP

```
asada@KB-PC-01-WIN10-1 MINGW64 ~ (main)
$ php -a
Interactive shell

php > echo -5%10.5;
PHP Deprecated:  Implicit conversion from
float 10.5 to int loses precision in php shell
code on line 1

Deprecated: Implicit conversion from float
10.5 to int loses precision in php shell code
on line 1
-5
php >
```

- In NODE/JS:

```
asada@KB-PC-01-WIN10-1 MINGW64 ~ (main)
$ node
Welcome to Node.js v18.14.2.
Type ".help" for more information.
> -5%10.5
-5
>
```

- In PYTHON:

```
asada@KB-PC-01-WIN10-1 MINGW64 ~ (main)
$ py
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7
2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or
"license" for more information.
>>> -5%10.5
```

```
5.5
>>>
```

** Different output, because Python does not automatically convert the floating-point number to integer, let’s try manually:

```
asada@KB-PC-01-WIN10-1 MINGW64 ~ (main)
$ py
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7
2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or
"license" for more information.
>>> -5%10
5
>>>
```

** Again, different output, in other languages, including my code the output for the remainder is -5 (negative), but in python the output for the remainder is 5 (positive).

In C, PHP, and NODE/JS, the modulo operation on a negative number results in a negative remainder. So, -5%10.5 (automatic conversion to -5%10) would result in -5, since -5 is the remainder when -5 is divided by 10.5.

In Python, the modulo operation is performed differently. The sign of the result is determined by the divisor (10.5 [manual conversion to 10] in this case), not the dividend (-5). Since 10 is positive, the result of -5%10 is positive, and the remainder is calculated accordingly, resulting in 5.

Whether it’s a negative number or a positive number the ultimate result of -5%10.5 is |5|, there is no chance to be -48, I think.

*** Seeking attention to review this code carefully

14. Program that will construct a menu for performing arithmetic operations. The user will give two real numbers (a, b) on which the arithmetic operations will be performed and an integer number (1 <= Choice <= 4) as a choice. Choice-1, 2, 3, 4 are for performing addition, subtraction, multiplication, division respectively. If Choice-4 is selected, the program will check if b is nonzero.

If the check is true, the program will ask for another choice (1 <= Case <=2), where Case-1, 2
evaluate quotient and reminder respectively. If the check is false, it will print an error
message “Error: Divisor is zero” and halt.

Code:

```
#include <stdio.h>

int main() {

    double a, b;
    int o, q_or_r;

    scanf("%lf %lf", &a, &b);
    scanf("%d", &o);

    switch(o) {
        case 1:
            printf("Addition: %g\n", a + b);
            break;
        case 2:
            printf("Subtraction: %g\n", a - b);
            break;
        case 3:
            printf("Multiplication: %g\n", a *
b);
            break;
        case 4:
            if (b != 0) {
                scanf("%d", &q_or_r);
                switch(q_or_r) {
                    case 1:
                        printf("Quotient:
%d\n", (int)a/(int)b);
                        break;
                    case 2:
                        printf("Reminder:
%d\n", (int)a%(int)b);
                        break;
                    default:
                        printf("Wrong
choice\n");
                }
            } else {
                printf("Error: Divisor is
zero\n");
            }
            break;
        default:
            printf("Wrong choice\n");
    }

    return 0;

} //main
```

Output:

➔ For input: 5 10
```
3
```

```
5 10
3
Multiplication: 50

Process returned 0 (0x0)   execution time : 4.434 s
Press any key to continue.
```

➔ For input: -5 10.5
```
4
2
```

```
-5 10.5
4
2
Reminder: -5

Process returned 0 (0x0)   execution time : 6.608 s
Press any key to continue.
```

➔ For input: -5 0

```
    4

-5 0
4
Error: Divisor is zero

Process returned 0 (0x0)   execution time : 5.977 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer* except for sample 2. In answer the reminder is `-48`, but from my code the output is `-5`, and I think there is no possible way to get `-48` for the input `-5 10.5`. For clearance, I tried the calculation in other languages (code added below) and compare those results with mine and the answer, from comparison I think my calculation and code contains no error, there is error in answer.

- In PHP

```
asada@KB-PC-01-WIN10-1 MINGW64 ~ (main)
$ php -a
Interactive shell

php > echo -5%10.5;
PHP Deprecated:  Implicit conversion from
float 10.5 to int loses precision in php shell
code on line 1

Deprecated: Implicit conversion from float
10.5 to int loses precision in php shell code
on line 1
-5
php >
```

- In NODE/JS:

```
asada@KB-PC-01-WIN10-1 MINGW64 ~ (main)
$ node
Welcome to Node.js v18.14.2.
Type ".help" for more information.
```

```
> -5%10.5
-5
>
```

- In PYTHON:

```
asada@KB-PC-01-WIN10-1 MINGW64 ~ (main)
$ py
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7
2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or
"license" for more information.
>>> -5%10.5
5.5
>>>
```

** Different output, because Python does not automatically convert the floating-point number to integer, let's try manually:

```
asada@KB-PC-01-WIN10-1 MINGW64 ~ (main)
$ py
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7
2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or
"license" for more information.
>>> -5%10
5
>>>
```

** Again, different output, in other languages, including my code the output for the remainder is `-5` (negative), but in python the output for the remainder is `5` (positive).

In C, PHP, and NODE/JS, the modulo operation on a negative number results in a negative remainder. So, `-5%10.5` (automatic conversion to `-5%10`) would result in `-5`, since `-5` is the remainder when `-5` is divided by `10.5`.

In Python, the modulo operation is performed differently. The sign of the result is determined by the divisor (10.5 [manual conversion to 10] in this case), not the dividend (-5). Since 10 is positive, the result of -5%10 is positive, and the remainder is calculated accordingly, resulting in 5.

Whether it's a negative number or a positive number the ultimate result of -5%10.5 is |5|, there is no chance to be -48, I think.

*** Seeking attention to review this code carefully

15. Program for "Guessing Game":
    Player-1 picks a number X and Player-2 has to guess that number within N = 3 tries. For each wrong guess by Player-2, the program prints "Wrong, N-1 Chance(s) Left!" If Player-2 successfully guesses the number, the program prints "Right, Player-2 wins!" and stops allowing further tries (if any left). Otherwise after the completion of N = 3 wrong tries, the program prints "Player-1 wins!" and halts.
    [Restriction: Without using loop/break/continue
    Hint: Use flag]

    Code:

```
#include <stdio.h>
#include <stdbool.h>

int main() {
    int x, guess, chances = 3;
    bool flag = false;

    scanf("%d", &x);

    scanf("%d", &guess);

    if (guess == x) {
        printf("Right, Player-2 wins!\n");
        flag = true;
    } else {
        printf("Wrong, %d chance(s) left!\n", --
chances);
    }

    if (!flag) {
        scanf("%d", &guess);
        if (guess == x) {
            printf("Right, Player-2 wins!\n");
            flag = true;
        } else {
            printf("Wrong, %d chance(s) left!\n", --
chances);
        }
    }

    if (!flag) {
        scanf("%d", &guess);
        if (guess == x) {
            printf("Right, Player-2 wins!\n");
        } else {
          printf("Wrong, %d chance(s) left!\n", --
chances);
            printf("Player-1 wins!\n");
        }
    }

    return 0;

} //main
```

Output:

➔ For input: 5
    12 8 5

```
5
12 8 5
Wrong, 2 Chance(s) Left!
Wrong, 1 Chance(s) Left!
Right, Player-2 wins!

Process returned 0 (0x0)   execution time : 9.133 s
Press any key to continue.
```

➔ For input: 100
    50 100

```
100
50 100
Wrong, 2 Chance(s) Left!
Right, Player-2 wins!

Process returned 0 (0x0)   execution time : 5.828 s
Press any key to continue.
```

➔ For input:
```
20
12 8 5
```

```
20
12 8 5
Wrong, 2 Chance(s) Left!
Wrong, 1 Chance(s) Left!
Wrong, 0 Chance(s) Left!
Player-1 wins!

Process returned 0 (0x0)   execution time : 10.671
s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

FILE: 9+99+999+.....+N.TXT

Code:

```
#include <stdio.h>

int main() {

     long int n, i, t = 9, sum = 0;

     printf("Input the number or terms: ");
     scanf("%ld",&n);

     for (i = 1; i <= n; i++) {
          sum += t;
          printf("%li\n", t);
          t = t*10+9;
     }
     printf("The sum of the series = %li\n", sum);

     return 0;

} //main
```

Output:

➔ For input: 5

```
Input the number or terms: 5
9
99
999
9999
99999
The sum of the series = 111105

Process returned 0 (0x0)   execution time : 6.468 s
Press any key to continue.
```

Observation:

Refactored given code just a little bit.

FILE: MULTIPLICATION USING LOOP.TXT

Code:

```c
#include <stdio.h>

int main() {
    int j, n;

    printf("Input the number (Table to be calculated): 
");
    scanf("%d",&n);

    for(j = 1; j <= 10; j++) {
        printf("%d X %d = %d \n", n, j, n*j);
    }

    return 0;

} //main
```

Output:

- ➔ For input: 5

```
Input the number (Table to be calculated): 5
5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
5 X 6 = 30
5 X 7 = 35
5 X 8 = 40
5 X 9 = 45
5 X 10 = 50

Process returned 0 (0x0)   execution time : 1.258 s
Press any key to continue.
```

Observation:

Refactored given code just a little bit.

FILE: SUM USING LOOP.TXT

Code:

```c
#include <stdio.h>

int main() {

     int i, n, sum = 0;

     printf("Input Value of terms: ");
     scanf("%d", &n);

     printf("The first %d natural numbers are:\n", n);

     for(i = 1; i <= n; i++) {
          printf("%d\t", i);
          sum += i;
     }

     printf("\nThe Sum of natural numbers upto %d terms:
%d\n", n, sum);

     return 0;

} //main
```

Output:

➔ For input: 5

```
Input Value of terms: 5
The first 5 natural numbers are:
1       2       3       4       5
The Sum of natural numbers upto 5 terms: 15

Process returned 0 (0x0)   execution time : 1.191 s
Press any key to continue.
```

Observation:

Refactored given code just a little bit.

FILE: CUBE USING LOOP.TXT

Code:

```
#include <stdio.h>

int main() {
     int i, ctr;

     printf("Input number of terms: ");
     scanf("%d", &ctr);

     for(i = 1; i<= ctr; i++) {
          printf("Number is: %d and cube of the %d is:
%d\n", i, i, (i*i*i));
     }

     return 0;

} //main
```

Output:

➔ For input: 5

```
Input number of terms: 5
Number is: 1 and cube of the 1 is: 1
Number is: 2 and cube of the 2 is: 8
Number is: 3 and cube of the 3 is: 27
Number is: 4 and cube of the 4 is: 64
Number is: 5 and cube of the 5 is: 125

Process returned 0 (0x0)   execution time : 4.990 s
Press any key to continue.
```

Observation:

Refactored given code just a little bit.

FILE: REVERSE A NUMBER.TXT

Code:

```
#include <stdio.h>

int main() {

     int n, reverse = 0, remainder;

     printf("Enter an integer: ");
     scanf("%d", &n);

     while (n != 0) {
          remainder = n % 10;
          reverse = reverse * 10 + remainder;
          n /= 10;
     }

     printf("Reversed number = %d", reverse);

     return 0;

} //main
```

Output:

➔ For input: 42957

```
Enter an integer: 42957
Reversed number = 75924
Process returned 0 (0x0)   execution time : 6.462 s
Press any key to continue.
```

Observation:

Refactored given code just a little bit.

FILE: 04. LOOP_RELATED_PROBLEMS.DOCX

Loop related problems (total 20 questions)

1. Write a program (WAP) that will print following series upto N th terms.
   1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, …….

Code:

```
#include <stdio.h>

int main() {

    int term;

    scanf("%d", &term);

    for(int i = 1; i <= term; i++) {
        if(i == term) {
            printf("%d", i);
            break;
        }
        printf("%d, ", i);
    }

    return 0;

} //main
```

Output:

➔ For input: 2

```
2
1, 2
Process returned 0 (0x0)   execution time : 6.138 s
Press any key to continue.
```

➔ For input: 5

```
5
1, 2, 3, 4, 5
```

```
Process returned 0 (0x0)   execution time : 5.461 s
Press any key to continue.
```

➔ For input: 11

```
11
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
Process returned 0 (0x0)   execution time : 1.413 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

2. Write a program (WAP) that will print following series upto N th terms.
   1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31 …….

Code:

```
#include <stdio.h>

int main() {

    int term, count = 0;

    scanf("%d", &term);

    for(int i = 1; term > 0; i += 2) {
        if(term == 1) {
            printf("%d", i);
            break;
        }
        printf("%d, ", i);
        --term;
    }

    return 0;

} //main
```

Output:

➔ For input: 2

```
2
1, 3
Process returned 0 (0x0)   execution time : 0.530 s
Press any key to continue.
```

➔ For input: 5

```
5
1, 3, 5, 7, 9
Process returned 0 (0x0)   execution time : 1.631 s
Press any key to continue.
```

➔ For input: 11

```
11
1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21
Process returned 0 (0x0)   execution time : 1.412 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

3. Write a program (WAP) that will print following series upto N th terms.
   1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, ...

Code:

```
#include <stdio.h>

int main() {

    int term, count = 0;

    scanf("%d", &term);

    for(int i = 1; i <= term; i++) {
        if(i%2 == 0) {
            printf("%d", 0);
```

```c
        } else {
            printf("%d", 1);
        }
        if (i != term) {
            printf(", ");
        }
    }

    return 0;

} //main
```

Output:

➔ For input: 1

```
1
1
Process returned 0 (0x0)   execution time : 1.063 s
Press any key to continue.
```

➔ For input: 2

```
2
1, 0
Process returned 0 (0x0)   execution time : 1.336 s
Press any key to continue.
```

➔ For input: 3

```
3
1, 0, 1
Process returned 0 (0x0)   execution time : 1.264 s
Press any key to continue.
```

➔ For input: 4

```
4
1, 0, 1, 0
Process returned 0 (0x0)   execution time : 0.953 s
Press any key to continue.
```

➔ For input: 7

```
7
1, 0, 1, 0, 1, 0, 1
Process returned 0 (0x0)   execution time : 1.265 s
Press any key to continue.
```

➔ For input: 13

```
13
1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1
Process returned 0 (0x0)   execution time : 1.459 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

4. Write a program (WAP) that will take N numbers as inputs and compute their average.
(Restriction: Without using any array)

Code:

```c
#include <stdio.h>

int main() {
    int n, i = 1;
    double num, avg, sum = 0;

    scanf("%d", &n);

    while (i <= n) {
        scanf("%lf", &num);
        sum += num;
        i++;
    }

    avg = sum / n;

    printf("AVG of %d inputs: %lf", n, avg);
    return 0;

} //main
```

Output:

➔ For input: 3
```
      10 20 30.5
```

```
3
10 20 30.5
AVG of 3 inputs: 20.166667
Process returned 0 (0x0)   execution time : 5.502 s
Press any key to continue.
```

➔ For input: 2
```
      22.4 11.1
```

```
2
22.4 11.1
AVG of 2 inputs: 16.750000
Process returned 0 (0x0)   execution time : 8.319 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

5. Write a program (WAP) that will take two numbers X and Y as inputs. Then it will print the square of X and increment (if X&lt;Y) or decrement (if X&gt;Y) X by 1, until X reaches Y. If and when X is equal to Y, the program prints “Reached!”

Code:

```
#include <stdio.h>

int main() {

    int x, y, square;

    scanf("%d %d", &x, &y);
    while (x != y) {
        square = x * x;
        printf("%d, ", square);
        if (x < y) {
            x++;
        } else {
            x--;
        }
    }

    printf("Reached!\n");

    return 0;
} //main
```

Output:

➔ For input: 10 5

```
10 5
100, 81, 64, 49, 36, Reached!

Process returned 0 (0x0)   execution time : 6.176 s
Press any key to continue.
```

➔ For input: 5 10

```
5 10
25, 36, 49, 64, 81, Reached!

Process returned 0 (0x0)   execution time : 5.576 s
Press any key to continue.
```

➔ For input: 10 10

```
10 10
Reached!

Process returned 0 (0x0)   execution time : 1.897 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

6. Write a program (WAP) for the described scenario:
   Player-1 picks a number X and Player-2 has to guess that number within N tries. For each wrong guess by Player-2, the program prints “Wrong, N-1 Choice(s) Left!” If Player-2 at any time successfully guesses the number, the program prints “Right, Player-2 wins!” and terminates right away. Otherwise after the completion of N wrong tries, the program prints “Player-1 wins!” and halts.
   (Hint: Use break/continue)

   Code:

```
#include <stdio.h>

int main() {

    int x, guess, n;

    scanf("%d", &x);

    scanf("%d", &n);

    for (int i = 1; i <= n; i++) {
        scanf("%d", &guess);

        if (guess == x) {
            printf("Right, Player-2 wins!\n");
            break;
        } else {
            printf("Wrong, %d choice(s) left!\n", n
- i);
        }
    }

    if (guess != x) {
        printf("Player-1 wins!\n");
    }

    return 0;
} //main
```

Output:

```
➔ For input: 5
             3
             12 8 5
```

```
5
3
12 8 5
Wrong, 2 choice(s) left!
Wrong, 1 choice(s) left!
Right, Player-2 wins!

Process returned 0 (0x0)   execution time : 5.728 s
Press any key to continue.
```

```
➔ For input: 100
             5
             50 100
```

```
100
5
50 100
Wrong, 4 choice(s) left!
Right, Player-2 wins!

Process returned 0 (0x0)   execution time : 7.555 s
Press any key to continue.
```

```
➔ For input: 20
             3
             12 8 5
```

```
20
3
12 8 5
Wrong, 2 choice(s) left!
Wrong, 1 choice(s) left!
Wrong, 0 choice(s) left!
Player-1 wins!

Process returned 0 (0x0)   execution time : 8.616 s
```

```
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

7. Write a program (WAP) that will run and show keyboard inputs until the user types an 'A' at the keyboard.

Code:

- First variant:

```
#include <stdio.h>
#include <stdbool.h>

int main() {

    int term = 1;
    char input;
    bool flag = true;

    while (flag) {
        scanf(" %c", &input);
        if (input == 'A') {
               flag = false;
               break;
        }
        printf("Input %d: %c\n", term, input);
        term++;
    }

    return 0;
} //main()
```

- Second variant:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

```
#include <stdbool.h>

int main() {

    int term = 0, inputs_size = 100, inputs_len = 0;
    char input;
    char* inputs = malloc(sizeof(char) * 100);
    char* part;
    bool flag = true;

    *inputs = '\0';

    for (int i = 0; flag; i++) {
        input = getchar();
        if(input == 'A') {
            flag = false;
            break;
        }
        if(input == '\n') {
               --i;
               continue;
        }
        term = i;
        char temp_input[20];
        sprintf(temp_input, "Input %d: %c,", term +
1, input);
        int temp_input_len = strlen(temp_input);
        if (inputs_len + temp_input_len >=
inputs_size) {
            inputs_size = inputs_len +
temp_input_len + 1;
            inputs = realloc(inputs, sizeof(char) *
inputs_size);
          }
        strcat(inputs, temp_input);
        inputs_len += temp_input_len;
    }

    part = strtok(inputs, ",");
```

```c
    while (part != NULL) {
        printf("%s\n", part);
        part = strtok(NULL, ",");
    }

    free(inputs);

    return 0;

} //main
```

Output:

** Both variant's output is same technically or theoretically but different in visually.

- First variant:
    - For input: X
      1
      a
      A

```
X
Input 1: X
1
Input 2: 1
a
Input 3: a
A

Process returned 0 (0x0)   execution time :
6.623 s
Press any key to continue.
```

- Second variant:
    - For input: X
      1
      a
      A

```
X
1
a
A
Input 1: X
Input 2: 1
Input 3: a

Process returned 0 (0x0)   execution time :
4.547 s
Press any key to continue.
```

Observation:

Though both variant gives the same output theoretically or technically, for second variant code, that gives the most similar output both theoretically or technically and visually with respect to answer.

Ultimately, no mismatch found between *Output* and *Answer*

8. Write a program (WAP) that will reverse the digits of an input integer.

Code:

```
#include <stdio.h>

int main() {

    int n, reverse = 0, remainder;


    scanf("%d", &n);

    while (n != 0) {
        remainder = n % 10;
        reverse = reverse * 10 + remainder;
        n /= 10;
    }

    printf("%d\n", reverse);
    return 0;

} //main
```

Output:

➔ For input: 13579

```
13579
97531

Process returned 0 (0x0)   execution time :
6.462 s
Press any key to continue.
```

➔ For input: 4321

```
4321
1234

Process returned 0 (0x0)   execution time :
6.462 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

9. Write a program (WAP) that will find the grade of N students. For each student, it will take the marks of his/her the attendance (on 5 marks), assignment (on 10 marks), class test (on 15 marks), midterm (on 50 marks), term final (on 100 marks). Then based on the tables shown below, the program will output his grade.

   ** Number distribution
   Attendance (A) 5%
   Assignments (HW) 10%
   Class Tests (CT) 15%

Midterm (MT) 30%
Final (TF) 40%

** Grade
90-100 A
86-89 A-
82-85 B+
78-81 B
74-77 B-
70-73 C+
66-69 C
62-65 C-
58-61 D+
55-57 D
Less than 55 F

Code:

```
#include <stdio.h>

int main() {

    int n;
    const double AT = 5.00, HWT = 10.00, CTT =
15.00, MTT = 50.00, TFT = 100.00;
    const double AP = 5.00, HWP = 10.00, CTP =
15.00, MTP = 30.00, TFP = 40.00;

    scanf("%d", &n);

    double as[n], hws[n], cts[n], mts[n], tfs[n],
totals[n], marks[n];
    char* grades[n];

    for (int i = 0; i < n; i++) {
        scanf("%lf %lf %lf %lf %lf", &as[i],
&hws[i], &cts[i], &mts[i], &tfs[i]);
        //totals[i] = (as[i] * (AP/100.0)) +
(hws[i] * (HWP/100.0)) + (cts[i] * (CTP/100.0)) +
(mts[i] * (MTP/100.0)) + (tfs[i] * (TFP/100.0));
        totals[i] = as[i] + hws[i] + cts[i] +
mts[i] + tfs[i];
        marks[i] = 
(totals[i]/(AT+HWT+CTT+MTT+TFT))*100.00;

        if (marks[i] >= 90 && marks[i] <= 100) {
             grades[i] = "A";
        } else if (marks[i] >= 86) {
             grades[i] = "A-";
        } else if (marks[i] >= 82) {
             grades[i] = "B+";
        } else if (marks[i] >= 78) {
             grades[i] = "B";
        } else if (marks[i] >= 74) {
             grades[i] = "B-";
        } else if (marks[i] >= 70) {
             grades[i] = "C+";
        } else if (marks[i] >= 66) {
             grades[i] = "C";
        } else if (marks[i] >= 62) {
             grades[i] = "C-";
        } else if (marks[i] >= 58) {
             grades[i] = "D+";
        } else if (marks[i] >= 55) {
             grades[i] = "D";
        } else if (marks[i] >= 0 && marks[i] < 55) 
{
             grades[i] = "F";
        } else {
             grades[i] = "Invalid";
        }
    }

    for (int i = 0; i < n; i++) {
        printf("Student %d: %s\n", i+1, 
grades[i]);
    }

    return 0;

} //main
```

Output:

➔ For input: 2
```
5 10 15 44.5 92.5
0 7.5 5 20 55.5
```

```
2
5 10 15 44.5 92.5
0 7.5 5 20 55.5
Student 1: A
Student 2: F

Process returned 0 (0x0)   execution time : 2.629 s
Press any key to continue.
```

Observation:

Though there is no mismatch found between *Question* and *Answer*, I myself not satisfied how I program this.

*** Seeking attention to review this code carefully

10. Write a program (WAP) that will give the sum of first N th terms for the following series.
1, -2, 3, -4, 5, -6, 7, -8, 9, -10, 11, -12, 13, -14, …….

Code:

```
#include <stdio.h>

int main() {

    int term, sum = 0;

    scanf("%d", &term);

    for (int i = 1; i <= term; i++) {
        if (i%2 == 0){
            sum += i*-1;
        }else{
            sum += i;
        }
    }

    printf("Result: %d\n", sum);

    return 0;

} //main
```

Output:

➔ For input: 2

```
2
Result: -1

Process returned 0 (0x0)   execution time : 0.959 s
Press any key to continue.
```

➔ For input: 3

```
3
Result: 2

Process returned 0 (0x0)   execution time : 1.145 s
Press any key to continue.
```

➔ For input: 4

```
4
Result: -2

Process returned 0 (0x0)   execution time : 0.919 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

11. Write a program (WAP) that will calculate the result for the first N th terms of the following series. [In that series sum, dot sign (.) means multiplication]
$1^2.2 + 2^2.3 + 3^2.4 + 4^2.5$

Code:

```
#include <stdio.h>
#include <math.h>

int main() {

     int term, sum = 0;

     scanf("%d", &term);

     for (int i = 1; i <= term; i++) {
          sum += pow(i, 2) * (i+1);
     }

     printf("Result: %d\n", sum);

     return 0;

} //main
```

Output:

➔ For input: 2

```
2
Result: 14

Process returned 0 (0x0)   execution time : 1.003 s
Press any key to continue.
```

➔ For input: 3

```
3
Result: 50

Process returned 0 (0x0)   execution time : 0.803 s
Press any key to continue.
```

➔ For input: 4

```
4
Result: 130

Process returned 0 (0x0)   execution time : 0.904 s
Press any key to continue.
```

➔ For input: 7

```
7
Result: 924

Process returned 0 (0x0)   execution time : 1.197 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

12. Write a program (WAP) that will print Fibonacci series upto N th terms.
    1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, …….

Code:

```c
#include <stdio.h>
#include <math.h>

int main() {

    int term, first, second, next;
    first = second = 1;

    scanf("%d", &term);

    for (int i = term; i > 0; i--) {
        if (i == 1) {
            printf("%d", first);
            break;
        }

        printf("%d, ", first);
```

```
            next = first + second;
            first = second;
            second = next;
        }

        return 0;

    } //main
```

Output:

➔ For input: 1

```
1
1
Process returned 0 (0x0)   execution time : 5.065 s
Press any key to continue.
```

➔ For input: 2

```
2
1, 1
Process returned 0 (0x0)   execution time : 5.099 s
Press any key to continue.
```

➔ For input: 4

```
4
1, 1, 2, 3
Process returned 0 (0x0)   execution time : 1.212 s
Press any key to continue.
```

➔ For input: 7

```
7
1, 1, 2, 3, 5, 8, 13
Process returned 0 (0x0)   execution time : 0.781 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

13. Write a program (WAP) that will print the factorial (N!) of a given number N. Please see the sample input output.

Code:

```c
#include <stdio.h>

int main() {

    int term, factorial = 1;

    scanf("%d", &term);

    printf("%d! = ", term);

    for (int i = term; i > 0; i--) {
          factorial *= i;
        if (i == 1) {
            printf("%d = ", i);
            break;
        }
          printf("%d X ", i);
    }

    printf("%d\n", factorial);

    return 0;

} //main
```

Output:

➔ For input: 1

```
1
1! = 1 = 1

Process returned 0 (0x0)   execution time : 0.982 s
Press any key to continue.
```

➔ For input: 2

```
2
2! = 2 X 1 = 2

Process returned 0 (0x0)   execution time : 0.619 s
Press any key to continue.
```

➔ For input: 3

```
3
3! = 3 X 2 X 1 = 6

Process returned 0 (0x0)   execution time : 0.852 s
Press any key to continue.
```

➔ For input: 4

```
4
4! = 4 X 3 X 2 X 1 = 24

Process returned 0 (0x0)   execution time : 0.093 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

I took help from the internet about how Fibonacci series works (https://www.mathsisfun.com/numbers/fibonacci-sequence.html).

14. Write a program (WAP) that will find ${}^{n}C_{r}$ where n &gt;= r; n and r are integers.

Code:

- First variant:

```
#include <stdio.h>

int main() {

    int total, choosen, total_factorial,
choosen_factorial, difference_factorial;
    total_factorial = choosen_factorial =
difference_factorial = 1;

    scanf("%d %d", &total, &choosen);

    for (int i = total; i > 0; i--) {
        total_factorial *= i;
    }

    for (int i = choosen; i > 0; i--) {
        choosen_factorial *= i;
    }

    for (int i = (total - choosen); i > 0; i--) {
        difference_factorial *= i;
    }

    printf("%d",
(total_factorial/(choosen_factorial*difference_facto
rial)));

    return 0;

} //main
```

- Second variant:

```
#include <stdio.h>

int factorial(int target) {
    int target_factorial = 1;
    for (int i = target; i > 0; i--) {
        target_factorial *= i;
    }
    return target_factorial;
}

int main() {
     int total, choosen;

     scanf("%d %d", &total, &choosen);

     printf("%d",
(factorial(total)/(factorial(choosen)*factorial(tota
l-choosen))));

     return 0;

} //main
```

Output:

** Both variants provide same output

➔ For input: 5 2

```
5 2
10
Process returned 0 (0x0)   execution time : 2.692 s
Press any key to continue.
```

➔ For input: 10 3

```
10 3
120
Process returned 0 (0x0)   execution time : 1.476 s
Press any key to continue.
```

➔ For input: 7 7

```
7 7
1
Process returned 0 (0x0)   execution time : 1.565 s
Press any key to continue.
```

➔ For input: 6 1

```
6 1
6
Process returned 0 (0x0)   execution time : 1.704 s
```

```
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

I took help from the internet about how $^{n}C_{r}$ can manually be calculate (https://www.cuemath.com/ncr-formula/).

15. Write a program (WAP) that will find x y (x to the power y) where x, y are positive integers.

Code:

```c
#include <stdio.h>
#include <math.h>

int main() {
     int x, y;
     double z;

     scanf("%d %d", &x, &y);

     z = pow(x, y);

     printf("%g", z);

     return 0;

} //main
```

Output:

➔ For input: 5 2

```
5 2
25
Process returned 0 (0x0)   execution time : 1.197 s
Press any key to continue.
```

➔ For input: 2 0

```
2 0
1
Process returned 0 (0x0)   execution time : 1.706 s
Press any key to continue.
```

➔ For input: 6 1

```
6 1
6
Process returned 0 (0x0)   execution time : 1.390 s
Press any key to continue.
```

➔ For input: 0 5

```
0 5
0
Process returned 0 (0x0)   execution time : 2.340 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

16. WAP that will find the GCD (greatest common divisor) and LCM (least common multiple) of two positive integers.

Code:

```
#include <stdio.h>

int main() {

    int x, y, gcd, lcm, temp, temp_x, temp_y;

    scanf("%d %d", &x, &y);

    temp_x = x;
    temp_y = y;

    if (x < y) {
        temp = x;
        temp_x = y;
        temp_y = temp;
    }
    while (temp_y != 0) {
        temp = temp_y;
        temp_y = temp_x % temp_y;
        temp_x = temp;
    }
    gcd = temp_x;

    lcm = (x * y) / gcd;

    printf("GCD: %d\n", gcd);
    printf("LCM: %d\n", lcm);

    return 0;

} // main
```

Output:

➔ For input: 5 7

```
5 7
GCD: 1
LCM: 35

Process returned 0 (0x0)   execution time : 1.445 s
Press any key to continue.
```

➔ For input: 12 12

```
12 12
GCD: 12
LCM: 12

Process returned 0 (0x0)   execution time : 2.041 s
Press any key to continue.
```

➔ For input: 12 32

```
12 32
```

```
GCD: 4
LCM: 96

Process returned 0 (0x0)   execution time : 1.677 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

I took help from the internet about how to find GCD and LCM in more details (https://www.idomaths.com/hcflcm.php).

17. WAP that will determine whether a number is prime or not.

Code:

```
#include <stdio.h>
#include <stdbool.h>

int main() {

    int num, factor = 0;
    bool flag = true;

    scanf("%d", &num);

    if(num == 0 || num == 1) {
          flag = false;
    }

    for(int i = 1; i <= num; i++) {
        if(num%i == 0) {
               ++factor;
        }
        if(factor > 2) {
               flag = false;
               break;
          }
    }
    if(flag) {
          printf("Prime\n");
    } else {
          printf("Not prime\n");
    }

    return 0;

} //main
```

Output:

➔ For input: 1

```
1
Not prime

Process returned 0 (0x0)   execution time : 1.003 s
Press any key to continue.
```

➔ For input: 2

```
2
Prime

Process returned 0 (0x0)   execution time : 1.148 s
Press any key to continue.
```

➔ For input: 11

```
11
Prime

Process returned 0 (0x0)   execution time : 1.378 s
Press any key to continue.
```

➔ For input: 39

```
39
Not prime

Process returned 0 (0x0)   execution time : 4.393 s
```

```
Press any key to continue.
```

➔ For input: 101

```
101
Prime

Process returned 0 (0x0)   execution time : 1.489 s
Press any key to continue.
```

Observation:

No mismatch between *Output* and *Answer*

18. WAP that will determine whether an integer is palindrome number or not.

Code:

```
#include <stdio.h>

int main() {

    int num, reversed = 0, remainder, original;

    scanf("%d", &num);

    original = num;

    while (num > 0) {
        remainder = num % 10;
        reversed = reversed * 10 + remainder;
        num /= 10;
    }

    if (original == reversed) {
        printf("Yes\n");
    } else {
        printf("No\n");
    }

    return 0;
} //main
```

Output:

- ➔ For input: 9

```
9
Yes

Process returned 0 (0x0)   execution time : 1.328 s
Press any key to continue.
```

- ➔ For input: 91

```
91
No

Process returned 0 (0x0)   execution time : 1.443 s
Press any key to continue.
```

- ➔ For input: 222

```
222
Yes

Process returned 0 (0x0)   execution time : 1.988 s
Press any key to continue.
```

- ➔ For input: 12321

```
12321
Yes

Process returned 0 (0x0)   execution time : 2.386 s
Press any key to continue.
```

- ➔ For input: 110

```
110
No

Process returned 0 (0x0)   execution time : 2.888 s
```

```
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*

I took help from the internet about what palindrome actually is, though I figured it out by analyzing the sample output (https://mathworld.wolfram.com/PalindromicNumber.html).

19. WAP that will calculate following mathematical function for the input of x. Use only the series to solve the problem.

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

Code:

```
#include <stdio.h>
#include <math.h>

int main() {


    char mode;
    double x, sin_x = 0.0;
    int n;

    scanf("%c", &mode);

    scanf("%lf", &x);

    scanf("%d", &n);

    if (mode == 'd') {
         x = x * M_PI / 180.0;
    }

    for (int i = 0; i <= n; i++) {
        int factorial = 1;
        for (int j = 2*i+1; j > 0; j--) {
            factorial *= j;
        }
        sin_x += pow(-1, i) * pow(x, 2 * i + 1) /
    factorial;
    }

    printf("%.3g\n", sin_x);

    return 0;

} //main
```

Output:

➔ For input: 1 (assuming x is in radius)

```
r
1
10
0.841

Process returned 0 (0x0)   execution time : 6.583 s
Press any key to continue.
```

➔ For input: 2 (assuming x is in radius)

```
r
2
10
0.902

Process returned 0 (0x0)   execution time : 5.219 s
Press any key to continue.
```

➔ For input: 3 (assuming x is in radius)

```
r
3
10
-19.7

Process returned 0 (0x0)   execution time : 5.327 s
```

```
Press any key to continue.
```

Observation:

First things first, the problem seems unclear, the equation is infinity and in question there is no instructions when or for how many terms the iteration should run, though I fixed it by manually asking that how many times the iteration should happen. Another problem is, there is also no hints that whether the value that the user will input is degree or rad. I also fixed that by asking from user. After fixing all problem program ran smoothly and correctly for rad input 1 and 2 (), but for input 3 I found mismatch between *Question* and *Answer*. I have no explanation for this problem.

*** Seeking attention to review this code carefully

20. Write a program that takes an integer number n as input and find out the sum of the following series up to n terms.

Code:

```
#include <stdio.h>

int main() {

    long int n, term = 1, sum = 0;

    scanf("%ld",&n);

    for (int i = 1; i <= n; i++) {
        sum += term;
        term = term*10+i+1;
    }
    printf("%li\n", sum);

    return 0;

} //main
```

Output:

➔ For input: 1

```
1
1

Process returned 0 (0x0)   execution time : 0.659 s
Press any key to continue.
```

➔ For input: 2

```
2
13

Process returned 0 (0x0)   execution time : 1.154 s
Press any key to continue.
```

➔ For input: 3

```
3
136

Process returned 0 (0x0)   execution time : 1.097 s
Press any key to continue.
```

➔ For input: 4

```
4
1370

Process returned 0 (0x0)   execution time : 1.147 s
Press any key to continue.
```

Observation:

No mismatch found between *Output* and *Answer*