**Rules**

1. Submission Steps:
   1. In your local machine, create a new folder; the name of the folder should be your 7 digit roll number.
   2. For each problem that you choose to solve, code your solution in a .c file. The name of the file should be exactly as is mentioned in problem description. Put this C file in the folder created in step 1(a).
   3. Finally, zip the folder created in step 1(a) to produce a .zip file. The name of the .zip file should be your 7 digit roll number.
   4. Submit the .zip file created in step 1(c).
2. You must submit your solution package (.zip file) by 10:00pm 14/06/2019 through Moodle.
3. You must bring your solution package in the next lab class (B1 🡪 15/06/2019, B2 🡪16/06/2019).
4. You must be able to explain your code properly in the lab class.
5. Do not copy code. You will be caught and given -20.

**Problem #1**

**Leap year**

**File name: leapyear.c**

In the Gregorian calendar, years that are multiples of 4 are leap years. However, exception to this rule occurs on the century year. A century year is a multiple of 100. Such a year is deemed as leap year only if it is a multiple of 400. Write a program that takes a year as input and outputs "Leap year" if the year is a leap year. Otherwise, it should output "Not a leap year". **You cannot use if-else or switch statements for this task**. (Hint: Use the *ternary operator* (?:))

|  |  |
| --- | --- |
| **Sample Input(s)** | **Corresponding Output(s)** |
| 1900  2000  1904  2004  2015 | Not a leap year  Leap year  Leap year  Leap year  Not a leap year |

**Problem #2**

**Accounting**

***File name: accounting.c***

In this problem, you are given the price of some commodity. For first 100 units, the rate is 0.5 Taka / unit. For the next 100 units, it is 0.8 taka / unit. Then onward, the price is 1.2 taka/unit. Your input is an integer representing the number of units bought by the customer. You must output a floating point number, upto 2 decimal digits, representing the total price.

|  |  |
| --- | --- |
| **Sample Input(s)** | **Corresponding Output(s)** |
| 100  200  300 | 50.00  130.00  250.00 |

**Problem #3**

**Max Count**

***File name: max.c***

In this problem, your input will be a positive number *n (0 < n < 10100)*. You need to output the number of occurrences of the largest digit in *n*. For example, in the first sample below, the output is 2, because the largest digit of 123322211 is 3 and 3 occurs twice in 123322211. (Input reading hint: You can read the input line character by character in a loop until you reach ‘\n’)

|  |  |
| --- | --- |
| **Sample Input(s)** | **Corresponding Output(s)** |
| 123322211  173625  8687557576676885  11111111111111111111111111111111  2  11223311122233 | 2  1  4  32  1  4 |

**Problem #4**

**Maclaurin series**

***File name: Maclaurin.c***

In this problem, your input is a real number *x,* which represents an angle in degree. You need to apply Maclaurin series expansion to compute the value of . Use the first 100 items from the infinite series. Print 6 digits after the decimal point in your output. Recall that, the expansion is as follows (where x is expressed in radian):

(Hint: Notice that the *n*th term is obtained from the *(n-1)*th term by multiplying with )

See the following example (minor precision error would be acceptable):

|  |  |
| --- | --- |
| **Sample Input(s)** | **Corresponding Output(s)** |
| 60  85  0  -210 | 0.500000  0.087156  1.000000  -0.866025 |

**Problem #5**

**Relatively Primes**

***File name: relprime.c***

Write a program that takes as input an integer *n* and prints the number of positive integers less than *n* that are relatively prime to *n*. Two integers *m* and *n* are relatively prime to each other if the greatest common divisor of them is equal to 1 i.e. gcd(m, n) = 1.

|  |  |
| --- | --- |
| **Sample Input(s)** | **Corresponding Output(s)** |
| 9 | 6 |

**Explanation:** There are 8 positive integers less than 9. Among them only the integers {1, 2, 4, 5, 7, 8} are relatively prime to 9. Thus there are 6 such integers and the answer is 6.