Assignment One – CIS*1500

#1

```
***********************************************************************************


      val = step % size; // value 3

      val = (450 - MAX)/ step; // value -16

      val = size % step;       // value 1

      len = PI * count;          //  value -3.14159

      val = step / -size;     // value 0

      len = step/size;       // value 0.0

      len = step % (step/size);// --> invalid because undefined value (cannot divide

3 by 0)! This will give an error when line is executed

      val = size/0;            //-->  invalid because undefined value (cannot divide

4 by 0)! This will give an error when line is executed

      val = step % (470-MAX); // value 3

      val = (MAX - 470)/size; // value 7

      len = size/count;       // value -4.0

      val = PI * step;     // --> invalid because the constant variable PI is a
double (floating point value) and the variable step is an integer value and when you
multiply a double by int the result is a double and so the holding variable must be a
type double, however the variable val is an integer. The solution to this is casting
val as a double to hold the resulting value.

      val = step % (MAX - 480) // --> invalid missing semicolon;

      val = (MAX - 480)%step; // value 2

      len = (double) step/size; // value 0.75

***********************************************************************************
```

#2

```
             int num1 = 0;

             double num2 = 28.6;

             char lett = ' ';


             printf("Enter an Int value:");
```

```
scanf("%d", &num1);

printf("\nEnter a character:");
scanf("%c", &lett);

printf("\nThe 3 values are %lf %d and %c", num2, num1, lett);
```

****

#3

```
int numSleep = 0;



printf("Enter the amount of sleep you had for the previous night");
scanf("%d" &numSleep);


if( numSleep >= 0 && numSleep <= 4)

{
        printf("Sleep deprived!");

}

else if(numSleep > 4 && numSleep < 6)

{
        printf("You need more sleep");

}

else if(numSleep >= 6 && numSleep < 8)

{
        printf("Not quite enough");

}


else if(numSleep >= 8)

{
        printf("Well Done!");

}
```

****

#4

```
double userNum = 0;

double sum = 0;

printf("Enter a value or enter -335 to exit");
```

```
        scanf("%lf", &userNum);

        while ((int)userNum != -335)

        {

            sum = sum + userNum;

            printf("\nEnter a value or enter -335 to exit");
            scanf("%lf", &userNum);



        }


        printf("\nSum of values: %.2f", sum);
```

****

#5

```
            char userChar = ' ';

            int a = 0;

            int b = 0;

            int c = 0;


    printf("Enter a next letter (a b or c) or type q to quit:");
    scanf("%c", &userChar);


    while(userChar != 'q')


    { //start of while

        switch(userChar)

        { //start of switch

        case 'a':
        case 'A':
                    a++;

                    printf("\na: %d\n", a );

                    break;
        case 'b':
        case 'B':
                    b++;

                    printf("\nb: %d\n", b );
```

```
                                 break;

             case 'c':
             case 'C':
                                 c++;

                                 printf("\nc: %d\n", c );

                                 break;


             default:
                          printf("\nOops! incorrect input");

                          break;


             }//end of switch


             printf("-- Please enter the next letter or type q to quit:");
             scanf("%c", &userChar);


       }//end of while
```

****

#6 Structured English

Declare and initialize array called monthLocation holding 12 integer values (each cell in the array represents 12 months).

Declare and initialize a corresponding array called amountDays holding 12 integer vales (each cell in the array represents the amount of days per month).

Declare and initialize a corresponding value called nameMonth holding 12 characters (each cell in the array represents the first letter of each month (e.g, ‘j’, ‘f’, ‘M’, a’, ‘m’).

User is asked to enter a year and is store into variable userYear.

A display of the letters representing each month is printed and User is asked to enter the first letter of a month (e.g. ‘j’ –- January , ‘f’ –-February etc) and is store into variable userNameMonth.

User is asked to enter the day of the month and is store into variable userDay.

Inside the body of a **FOR** loop, the variable userNameMonth (e.g 'm') is compared to each cell in the array nameMonth (j, f, M, a), to find the location of the user's month within the 12 month period.

Ex, if the user enters 'M' (March), the loop will continue until x finds the index of 'M', corresponding to the array called monthLocation (1st month, 2nd month, 3rd month etc) (parallel arrays).

Variable called LocationSet stores the index of 'M'

THEN, a method is used to calculate the total amount of days in the leap years and regular years, using a **FOR** loop.

Variable x is initialized to 2000 and increments by 1 until x is less than userYear. e.g (user enters 2005, loops stops iterating until 2004).

Inside the body of a new **FOR** loop, **IF the remainder of (x (e.g 2000)/4) equals 0**, then 366 days is multiplied by the variable Y (initialized to 1) –-> represents the amount of occurrences of a leap year.

Y is incremented by 1

ELSE 365 days is multiplied by Z (initialized to 1) –-> represents the amount of occurrences of a normal year.

Z is incremented by 1

AFTER the condition is met (less then userYear) the amount of total days from 2000 to 2004 is calculated by adding both the accumulated days in the leaps years and accumulated days in the normal years , stored in variable called totalDays.

THEN, a method is used to calculate the total amount of days since the date Jan 1 2000

IF userYear equals 2000 and userNameMonth does not equal 'j' (represents Jan), inside the condition of a new **FOR** loop, the variable x is initialized to 1 and increments by 1 until its x is less than the variable locationSet (month's location year, e.g 1st 2nd 3rd).

Variable daysFromYear (initialized to 0) equals daysFromYear plus the amount of days in each month –-> array called amountDays (e.g 30, 31, 31) (index increments by 1 after each iteration).

Variable daysFromYear is incremented by 1 because the year 2000 as part of the **IF** condition, is a leap year

ELSEIF userYear equals 2000 and userNameMonth equals 'j', then daysFromYear equals userDays (day of month user entered) plus 1 –-> because leap year occurs.

ELSEIF userYear is greater than 2000 and userNameMonth does not equal ‘j’.

FOR Loop is executed where inside the condition of a new **FOR** loop, the variable x is initialized to 1 and increments by 1 until its x is less than the variable locationSet (month’s location in year, e.g 1st 2nd 3rd).

Variable daysFromYears equals totalDays (calculated from the sum of the accumulated days in the leaps years and accumulated days in the normal years) plus the amount of days in each month –-> array called amountDays (e.g 30, 31, 31) (index increments by 1 after each iteration).

IF the remainder of (userYear/4) equals 0, then variable daysFromYear equals (DaysFromYear plus userDays) plus 1

ELSE variable daysFromYears equals (DaysFromYear plus userDays) minus 1

ELSEIF userYear is greater than 2000 and userNameMonth equals ‘j’

IF the remainder of (userYear/4) equals 0, the variable daysFromYear equals userDays plus 1

ELSE the variable daysFromYear equals userDays minus 1

A statement prints the amount of days since January 1 2000 using the variable daysFromYear.