APS 105: Computer Fundamentals

Tutorial #2
Summer 2025

Problem 1: Math Library 1(Modified from Fall 2014, Q3)

Write a single C statement that will compute the value of r according to the following expression, and assign the value to r.

$$r = \frac{x^n + 6 \times x^4}{\sin(y) + \cos(z)} \tag{1}$$

Example solution:

```
r = (pow(x, n) + 6 * pow(x, 4.0)) / (sin(y) + cos(z));
```

Problem 2: Math Library 2 (Winter 2023, Q1)

Write a single C statement that adds the square root of each digit in a two-digit number num. In the same statement, declare and assign the result to a double variable named square. For example, if num has 64, you should declare and assign to square $\sqrt{6} + \sqrt{4}$. You can use *sqrt()* function from math.h library

Example solution:

```
double square = sqrt(num / 10) + sqrt(num % 10);
```

Problem 3: Random Number Generation 1 (Fall 2018, Q2)

Write a single C statement that generates a random even number in the range of $[-150, 150]$ (inclusive), and uses it to declare and initialize an int-type variable *randomChoice*.

Example solution:

```
int randomChoice = (rand() % 151 - 75) * 2;
```

Problem 4: Random Number Generation 2 (Winter 2024, Q3)

Write a single C statement that declares an int variable called randomNum and sets it to a random number that is a multiple of 5 and is between 5 and 25 (inclusive). This means that randomNum should be set to either: 5, 10, 15, 20 or 25. You may use the *rand()* function.

Example solution:

```c
int randomNum = ((rand() % 5) + 1) * 5;
```

Problem 5: Estimate e (Winter 2020, Q6)

The value of the mathematical constant e can be expressed using the infinite series:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \dots \tag{2}$$

Write a C program that approximates the value of e.

Rather than adding an infinite number of terms, your program should continue adding terms until the value of a term is less than 0.001. Your program should print the approximation to e and the number of terms used to determine the approximation. The terms in the series are $1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!}$and so on.

Example solution:

```c
#include <stdio.h>

int main(void) {

  const double TOLERANCE = 0.001;
  double sum = 0.0, term = 1.0;
  int n = 0;

  while (term >= TOLERANCE) {
    sum = sum + term; // accumulate the term
    n = n + 1; // determine next term
    term = term / n;
  }
  printf("The value of e is approximately %f.\n", sum);
  printf("The number of terms in the sum is %d.\n", n);
  return 0;
}
```

Problem 6: Print Consecutive Digits

Write a complete C program that first prompts the user to enter a floating point number between 0 and 10, and its number of digits after the decimal point (no greater than 7). It

will then print all possible two-digit numbers found in consecutive digits of the floating point number that the user entered. For example, if the number entered is 2.7182818, and the number of digits entered is 7, the two-digit numbers printed are:

```
Enter a floating point number between 0 and 10: 2.7182818
Enter the number of digits after the decimal point: 7
27
71
18
82
28
81
18
```

Example solution:

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

int main(void) {

  double input;
  int digits;

  printf("Enter a floating point number between 0 and 10: ");
  scanf("%lf", &input);
  printf("Enter the number of digits after the decimal point: ");
  scanf("%d", &digits);

  for(int i = 0; i < digits; i++) {

    int p1 = (int)(input * pow(10, i)) % 10;
    int p2 = (int)(input * pow(10, i+1)) % 10;
    printf("%d%d\n", p1, p2);
  }
return 0;
}
```

Problem 7: Simple Function (Winter 2018, Q4)

Write a function in C called *mostSignificantDigit*, that returns the most significant digit of a positive int- type integer that is passed to the function. For example, if the function is called with the value 987654321 as its argument, it will return 9.

Example solution:

```c
int mostSignificantDigit(int number) {

  int leadingDigit;

```

```
  while (number > 0) {
    leadingDigit = number % 10;
    number /= 10;
  }
return leadingDigit;
}
```