Exercise Problems - 10

Functions - 1

1. Write a program that accepts a number n from the user and prints all prime numbers less than n, by repeatedly calling the function `isprime()` we discussed in class.

2. Write a program that takes n numbers from the user and stores them into an array and then computes the maximum of the elements of the array by repeatedly calling the function `maximum()` that we discussed in class. Recall that this function takes two integers as parameters and returns an integer value.

3. In this exercise, we will develop a program that approximately computes $cos(x)$ (where x is in radians), using the formula $cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$. Write a C function `cos` that takes two parameters, a float variable x and a positive integer k and returns a double value which is an approximation of $cos(x)$ obtained by summing up the first k terms of the above series expansion. For example, if $x = 1.5$ and $k = 1$, the return value should be 1 and if $x = 1.5$ and $k = 3$, the return value should be the resullt of $1 - \frac{1.5^2}{2!} + \frac{1.5^4}{4!}$. Write a main function that accepts a value x and number of terms n as inputs from the user, invokes the function $cos()$ with parameters x and n outputs the resultant value to the user. Note that, cos() function will be more efficient if you compute every term in the series from its previous term, than recompute each term from the beginning.

4. Write a program that takes a positivr number n and a non-negative number k as inputs from the user and outputs the value of nP_k. To implement this, write a function `CountPermut` which takes two parameters n and k and computes nP_k using the formula

   $$nP_k = n \times (n-1) \times \cdots (n-k+1)$$

   and returns this value. Your main function should take inputs from the user, do necessasary validations and call the function `CountPermut` to compute the result, and then outputs the result to the user.

5. Rewrite the above program by redefining the `CountPermut` function to be a recursive function, which uses the formula:

   $$nP_k = 1, \text{ if } k = 0 \text{ and } nPk = nP_{k-1} \times n - k + 1, \text{ otherwise.}$$

   Make sure that your function terminates for all possible combinations of input values.