MTH 3300
E. Fink

3300 Problems, Section 6: Functions

1. Write a C++ function which takes two `doubles` x and y as input, and returns $\sqrt{x^2+y^2}$ as output. (2 lines)
2. Write a user-defined function representing $f(x) = |x|^3$.
3. Write C++ a function representing the mathematical function $f(x) = x^2 + 1$. (So, for example, I could write `cout << f(2.1);` and `5.41` would appear on the console.) (2 lines)
4. Two planes are a dangerous distance from one another if the difference between their heights is less than 2000 feet. Given that, write a function `too_close` that makes the following code work:

```
double height1, height2;
cout << "Enter heights of the two planes:  ";
cin >> height1 >> height2;
if(too_close(height1, height2))
{
      cout << "Too close!";
}
else
{
      cout << "Ok";
}
```

   (2 lines, although more is definitely ok.)
5. Write the **function header** for a function named `grade` which has 3 inputs: a double containing an assignment score, an integer corresponding to assignment number, and a string corresponding to a name. The function will return a letter grade. DO NOT write the function itself. (1 line)
6. Write a function that has one `int` argument, and returns nothing but results in `Even!` being printed to the console if the input is even and `Odd!` being printed to the console if the input is odd. (5 lines)
7. Create a function named `myNum` which has 0 inputs, and which returns a `double` number accepted from the keyboard. (4 lines)
8. Describe the crucial error in this function (in one sentence or less):

```
void fff(int x, int y, int &z)
{
      z = x + y;
      return z;
}
```

9. Explain the error in the following function (1 sentence or less):

```
double letter(char x, char y, int z)
{
      if(z <=10)
      {
            cout << x << endl;
      }
}
```

10. Write a function called `digitX`. This function should have an `int` as input, and it should not return anything. The function should **print out** the input value if it is a single digit number (that is, `0` through `9`); and it should **print out** the character `x` if the input value is *not* single digit. (5 lines; read the problem carefully!)
11. Write a function called `vowelCensor`. This function should have a `char` as input, and it should not return anything. The function should **print out** the input value if it is *not* a lowercase vowel (that is, `a`, `e`, `i`, `o` and `u`); and it should **print out** the character `!` if the input value *is* a lowercase vowel. (5 lines)
12. Consider the following portion of a program:

```
int f(int &a)
{
      cout << a << endl;
      a++;
      return a + 2;
}

int main()
{
      int x = 2, y;
      y = f(x);
      cout << x << endl;
      cout << y << endl;
      return 0;
}
```

What will print out?

13. Consider the following portion of a program:

```
void f(int &a, &b)
{
      cout << a << endl;
      b = a;
}

int main()
{
      int x = 2, y = 3;
      f(x,y);
      cout << x << endl;
      cout << y << endl;
      return 0;
}
```

What will print out?

14. Consider the following portion of a program:

```
int blah(int &a, int b)
{
      a++;
      b++;
      return a+b;
}

int main()
{
      int a = 2, b = 5, c;
      c = blah(a,b);
      cout << a << endl;
      cout << b << endl;
      cout << c << endl;
      return 0;
}
```

What will print out?

15. Determine what the following code displays.

```
int fn(int x, int &y)
{
      x = x+1;
      y = y+2;
      return x+y;
}

int main()
{
      int a = 10, b= 20;
      int c = fn(a,b);
      int d = fn(b,c);
      cout << a << endl << b << endl << c << endl << d;
      return 0;
}
```

16. Write the function HEADING for a function which has one integer and one string as "input" and "returns" two characters. (You can name the function and parameters anything you like.) Don't write the function itself! (1 line)

17. Describe TWO reasons to use reference parameters when defining functions.

18. Write a function named `replaceMax` which takes two `int`s as arguments, and returns nothing. The function should change the values of the arguments so that after the function call, both of the arguments are equal to the greater of the two (if they are the same, the new values for each should be the same as the old ones). For example, the following code should print out `5 5`:

    ```
    int a = 3; b = 5;
    replaceMax(a,b);
    cout << a << " " << b;
    ```

    (5 lines)

19. Write a function called `randomChange`. This function will take two `char`s as input, and return nothing. The function will randomly choose one of the two variables, and change that variable's value to `'!'`. The *actual* parameter's value should change – that is, if I call `randomChange(x, y)` in `main()`, then either the value of `x` or the value of `y` should change in `main()`. You may omit the seeding line. (5 lines)

20. Write a function that takes in four inputs, representing the x- and y-coordinates of two points. The function should "return" (in the manner described in class) the x and y coordinates of the **midpoint** of the two entered points. (Recall that to get the midpoint of two points, you average their x-coordinates and average their y-coordinates.) (3 lines, the first is pretty long)

21. Write the HEADING of a C++ function that takes three `int`s as input and "returns" (in the manner we described in class) two outputs: the highest of the three, and the lowest of the three. Don't write the function itself!

22. Determine what the following code displays (be sure to write out the answer as it will shown on the console!). Don't worry about `system("pause");`.

```
#include <iostream>
using namespace std;

int x = 10;

int fn(int y)
{
      x = y;
      cout << y << endl;
      return x+y;
}

int main()
{
      cout << x << endl;
      int x = 5;
      x = fn(x);
      cout << x << ::x << endl;
      return 0;
}
```

23. What will print out from this program?

```
#include <iostream>
using namespace std;

char letter = 'c';

char encode(int k, char &y)
{
      y +=k;
      return y;
}

int main()
{
      cout << letter;
      char letter = 'n';
      char coded = encode(2, letter);
      cout << ::letter << coded << letter;
      return 0;
}
```

24. What will be displayed by the following?

```
#include <iostream>
using namespace std;

string x = "Hello";

int main()
{
      cout << x << endl;
      string x = "Goodbye";
      cout << x << endl << ::x << endl << "x" << endl;
      return 0;
}
```

25. Determine what the following code displays (be sure to write out the answer as it will shown on the console!). Don't worry about `system("pause");`.

```
#include <iostream>
using namespace std;

int x = 10;

int fn(int x)
{
      int y = 2;
      x = y;
      return x+y;
}

int main()
{
      cout << x << endl;
      int x = 5, y = 3;
      y = fn(::x);
      cout << x << y << endl;
      return 0;
}
```

Longer Problems

1. Recall the formula $e^x \approx 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!}$, where n can be any integer greater than 1. The higher the value of n, the more accurate your estimate is.

   Write a program that asks the user to input a value of x, and a power n, that outputs an approximation to e^x using the terms up to power n. For example, if the user enter `0.1` for x and `3` for n, the program should output `1.10517`, because $1 + \frac{0.1}{1!} + \frac{0.1^2}{2!} + \frac{0.1^3}{3!} = 1.10517$ (when rounded).

   For full credit, make a function called `myExp` that computes the approximation: this function should have two arguments, a `double` named `x` and an `int` named `n`, and should return the approximate value of e^x using the terms up to that power. IMPORTANT: you may assume that a function with the prototype `int fact(int)` has been defined, which computes factorials; you DO NOT have to write this function, but you may use it. (≈10-15 lines)

2. Write the following program. You may leave out `#include`s.

   The program should open a file named `news.txt` and print out its content to the console, except with every appearance of the words "`Hillary`" or "`Donald`" replaced with "`***`". So, for example, if `news.txt` contained the sentence

   ```
   Hillary and Donald ate lunch together in the park
   ```

   then the sentence that prints out to the console should be

   ```
   *** and *** ate lunch together in the park
   ```

   (You don't need to worry about capitalization, punctuation, or newlines – but do assume that there is a space between each word!)

   For full credit, write and use a function called `newword` that takes a `string` as input, and returns either the `string` `***` (if the input is `Hillary` or `Donald`) or the original input (if the input is any other `string`). So, `newword("Hillary")` should return "`***`", while `newword("park")` should return "`park`". (≈ 13 lines)

3. Write the following program. You may leave out `#include`s. (≈ 11 lines)

   A **pyramidal** number is a number that is the sum of the first n consecutive square integers. So, the first pyramidal number is 1 (since $1^2 = 1$); the second pyramidal number is 5 (since $1^2 + 2^2 = 5$); the third pyramidal number is 14 (since $1^2 + 2^2 + 3^2 = 14$); and so on.

   Write a program that allows the user to input an integer n, and the prints out the **first n pyramidal numbers.** For example, if the user enter `3`, the program would print out the first 3 pyramidal numbers: `1`, `5`, and `14`, each on a different line.

   For full credit, write and use a **function** called `pyr` whose input is an integer `x`, and whose output is the `x`th pyramidal number. For example, the value of `pyr(3)` should be 14.

   Your program and function only need to work if the inputs are positive integers.

4. Write the following program. You may leave out `#include`s.

   The program should ask the user to enter integer values n and k, and then compute n choose k (you may have seen this written as ${}_nC_k$ or $\binom{n}{k}$), which is given by the formula $\frac{n!}{k!(n-k)!}$. For example, ${}_5C_2 = \frac{5!}{2!3!} = \frac{120}{2 \cdot 6} = 10$.

   For full credit, use a **function** as part of your program – you probably want to write the factorial operation as a function; otherwise, you will need several loops, instead of just one!

   Your program only needs to work if the entered n and k values are 0 or positive. (≈ 11 lines)

5. Write a *whole* program that does the following: The program should create 8-letter passwords, made up of 8 random lowercase letters, until the user is satisfied.

   Specifically, it should do the following repeatedly: first, it should print out a random password. Then, it should ask the user if the password is satisfactory. If the user enters `y`, the program ends; if the user enters `n`, everything repeats. (You may assume the user enters only `y` or `n`.)

   For full credit, you should use a FUNCTION that takes NO arguments, and returns a `string` (the random password).

   Hints: to create a random lowercase letter, add a random number to the character `'a'`. To create the full random password, start with an empty `string`, and `+=` random lowercase letters on to the end, one letter at a time.

   Also, your first line of `main()` should be `srand(time(NULL));` You might also want to declare a `bool` outside your loop in `main()` – that's just a suggestion, there are other ways to go.

(≈ 5 lines for the function, ≈ 8 lines in `main()`)

6. Two words are called *granagrams* if they have the same number of `g`'s AND the same number of `r`'s in them. For instance, `greening` and `reigning` are granagrams, because they each have two `g`'s and one `r`. `ggrrrh` and `rrxkabgrg` *are* granagrams, while `rage` and `garbage` are *not*.

   Write a *whole* program that does the following: first, it asks the user to enter two words. Then, the program will print `Granagrams` or `Not Granagrams` depending on whether those two words are granagrams. ASSUME that all letters entered are LOWERCASE.

   For full credit, you should use a FUNCTION – or perhaps a couple, your choice – which returns a count of the number of appearances of a certain letter in a `string`. So your function(s) should have at least one argument which is a `string`, and should return an `int` – beyond that is up to you. (≈ 7 lines in `main()`, and ≈ 6 lines in the function – I wrote one function)

7. A *palindrome* is a number that reads the same forwards and backwards, for example `121` or `2442`. Write a program that checks whether an integer entered by the user is a palindrome. Example run:

   ```
   Enter an integer:  2442
   That's a palindrome!
   ```

   or:

   ```
   Enter an integer:  14
   Nope, not a palindrome.
   ```

   (In these examples, `2442` and `14` are user input.)

   You must do it as follows for full credit: first, have the user enter the integer, which should be stored AS A STRING. Then, you should write a function called `reverse`, which takes in a `string` as an argument, and then returns a new `string` which is the input reversed. (So, for example, if `string x = "Hello"`, then when I write `cout << reverse(x);` it would print out `olleH`.) You should also use this function!

   Hints: write the `main()` function first, hopefully that is easy, and worth substantial partial credit. Then write the function. In the function, you should start with a new empty `string` for output, and then add on to it one character at a time. (≈10 - 15 lines)