Homework #2

Question 1 (5 pt.)

The following C function calculates the Euclidean distance between two points (or pixels) in a two-dimensional space, where each pixel is represented through a data structure named `Pixel`, defined as shown below.

```
struct Pixel {
    double x;
    double y;
};

double getPixelDistance(struct Pixel *pixels)
{
    double dist_x = pixels[1].x - pixels[0].x;
    double dist_y = pixels[1].y - pixels[0].y;
    return sqrt(dist_x * dist_x + dist_y * dist_y);
}
```

From a main program, function `getPixelDistance()` could be invoked as follows:

```
struct Pixel pixels[2];
pixels[0].x = 1.0;
pixels[0].y = 2.5;
pixels[1].x = 4.0;
pixels[1].y = 3.0;
double dist = getPixelDistance(pixels);
```

Write an LLVM implementation for function `getPixelDistance()`, and upload your code in a file named `q1.ll`. No other file is required, but it is recommended that you write a main program to test the correctness of your code.

Notice that there is a call to function `sqrt()` in the body of `getPixelDistance`. The implementation of this function is available in the math library (added with flag `-lm` in the *gcc* linker). In order to invoke it from your LLVM code, you only need a previous declaration of its prototype in LLVM format with the following line of code:

```
declare double @sqrt(double)
```

After declared, this function can be normally invoked with the `call` instruction, as seen in class.

Question 2 (5 pt.)

Consider a string comparison function with the following prototype:

```
int StringCompare(char *s1, char *s2);
```

The function takes two null-terminated string as its arguments and returns -1, 0, or 1 if the first string is less than, equal to, or greater than the second, respectively. A sting `s1` is considered to be *less than* a string `s2` if the first character that differs in both strings has a lower ASCII code in `s1`.

a) Write the body of functiøn `StringCompare()` in C. Write a comment next to each line of code that marks the beginning of a new basic block, as assign a label to it. These labels must match the basic block names in the LLVM code submitted for the following question. Upload the C code in a file named `q2.c`.

b) Write an equivalent version of this function in LLVM. Name temporary variables as `%t0`, `%t1`, etc., and write a comment for each line of code indicating the type of each new variable, and a description of its assigned value using C-like psuedo-code. Upload this code in a file named `q2.ll`.