Question 1

In lecture, we saw that in order to run a C program (e.g., `hello.c`), we first need to run the command `make hello`, and then run the command `./hello`.

a) What does running `make hello` do?
b) What does running `./hello` do?
c) What might happen if you were to run `./hello` without first running `make hello`?

Answers

a) TODO
b) TODO
c) TODO

Question 2

In your own words, what does it mean for a function to have

a. arguments?
b. a return value?
c. side effects?

For each of (a), (b), and (c), give one example.

Answer

a. TODO
b. TODO
c. TODO

Question 3

Recall that, in lecture, we saw the following two blocks of code, both of which print the same output.

Version 1

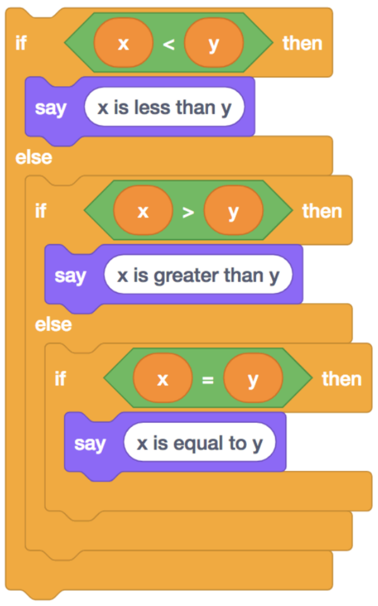
```
if (x < y)
{
    printf("x is less than y\n");
}
else if (x > y)
{
    printf("x is greater than y\n");
}
else if (x == y)
{
    printf("x is equal to y\n");
}
```

Version 2

```
if (x < y)
{
    printf("x is less than y\n");
}
else if (x > y)
{
    printf("x is greater than y\n");
}
else
{
    printf("x is equal to y\n");
}
```

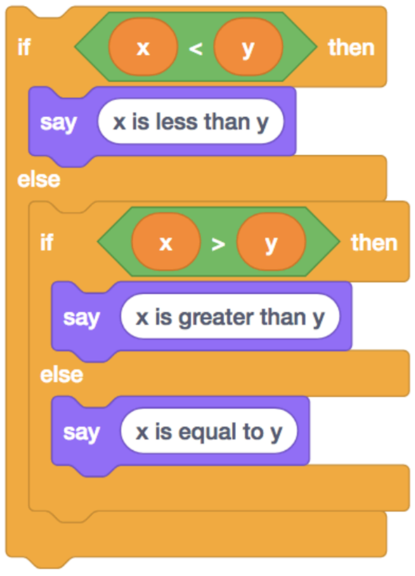
These are really just the C equivalents of the following two blocks of Scratch code.

Version 1

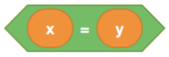


```
if x < y then
  say x is less than y
else
  if x > y then
    say x is greater than y
  else
    if x = y then
      say x is equal to y
```

Version 2



```
if x < y then
  say x is less than y
else
  if x > y then
    say x is greater than y
  else
    say x is equal to y
```

a) Why, in C, do we use two equals signs (`==`) when we write `else if (x == y)`, whereas in Scratch we use just a single equals sign (`=`) in 

?
b) Why is Version 2 of the code, whether implemented in Scratch or in C, arguably better designed than Version 1?

Answers

a) TODO
b) TODO