CSCI 1300: Assignment #1

Due on January 27, 2017 at 12:30pm

Professor David Knox Section 100

John Keller

1. Using those three rates of change, and a current U.S. population of 318,933,342, write a program to calculate the U.S. population in exactly one year (365 days). Your algorithm should output the result of your calculations.

```
// Test results of calculations are commented at end of each line
current_population = 318933342;
days_past = 365;
immigrants = 24 * 60 * 24 * days_past; // 12614400
deaths = 2 * 60 * 24 * days_past; // 1051200
births = 8 * 60 * 24 * days_past; // 4204800
final_population = immigrants + births + current_population - deaths; //
    ↪ 334701342
output(final_population);
```

2. A day has 86,400 seconds (24*60*60). Given a number of seconds in the range of 0 to 86,400, output the time as hours, minutes, and seconds for a 24 hour clock. For example, 70,000 seconds is 19 hours, 26 minutes, and 40 seconds. Your program should have user input that is the number of seconds to convert, and then use that number in your calculations. Your output should be displayed as The time is X hours, Y minutes, and Z seconds.

```
seconds = input("Enter the number of seconds you want to convert!");
if seconds less than 86400 // Error handling
    output("Error: You entered more seconds than are in a day!");
else // No error; let's run the calculations
    // The Math.floor() command removes the decimals from each number
    hours = Math.floor(seconds / 86400 * 24);
    minutes = Math.floor(((seconds / 86400 * 24) - hours) * 60);
    seconds = Math.floor((((seconds / 86400 * 24) - hours) * 60 - minutes) *
        ↪ 60);
    output("The time is "+hours+" hours, "+minutes+" minutes, and "+seconds+"
        ↪ seconds");
    // The plus signifies that the numbers and strings are combined for the
        ↪ human readable output
```

3. In science, temperature is always described in Celsius, but in the U.S. we tend to use Fahrenheit temperatures. Write an algorithm to convert a Celsius temperature into Fahrenheit. (Subtracting thirty two from the Fahrenheit value and taking five ninths of that result will provide the Celsius value)

```
c = input("Enter the degree C you want to convert!");
f = (c-32)*(5/9); // Subtracts 32, then multiplies it by 5/9.
output(f);
```

4. Write an algorithm that asks a user to enter a number between 1 and 10. (This range includes the numbers 1 and 10.) When they enter the number, check that it is actually between 1 and 10. If it is not, ask them to enter a number again. Continue to ask until they enter a valid number. Once their number is valid, output the number.

```
number = input("Enter a number between 1 and 10");
while number greater than 10 or less than 0
    number = input("Error: Enter a number between 1 and 10"); // Let's try to
        ↪ get a number between 1 and 10 again
output(number);
```

5. Write an algorithm that asks a user the miles per gallon of their car. If they say greater than 30, output Nice job. If they say between 15 and 29, output Not great, but okay. If they say less than 15, output So bad, so very, very bad.

```
mpg = input("Enter the number of miles per gallon for your car");
if mpg greater than or equal to 30
    output("Nice job");
else if mpg greater than 15
    output("Not great, but okay");
else
    output("So bad, so very, very bad.");
```

6. In text-based choose your own adventure games, the game player is presented with choices throughout the game and then the game responds based on the users choice. Write the algorithm for a simple choose your own adventure game where the user has three choices:

   - Fight the dragon
   - Go home
   - Save the princess

   The game should repeatedly ask the user which of the three options they want to do until the user says Go home. When Go home is selected, the loop should exit, which effectively ends the game.

   If the user selects Fight the dragon, the algorithm should output You win!. If Save the princess is selected, the algorithm should output You saved the princess. If Go home is selected, the algorithm should output Wimp.

   You can set up your algorithm to check for the users input in any way you like. Checking for the actual words, such as Go home is one option. If you want to assign a number to each option and check for the number, that also works.

```
// Note: The instructions for this problem were very confusing, I did the
    ↪ while to the best of my ability.
game_choice = input("Enter the game you want to play: Fight the dragon, Go
    ↪ home, or Save the princess");
while game_choice is not "Go home"
    game_choice = input("Enter the game you want to play: Fight the dragon, Go
        ↪  home, or Save the princess");
output("The game is effectively over.");
game_choice = input("Enter the game you want to play: Fight the dragon, Go
    ↪ home, or Save the princess");
if game_choice is "Fight the dragon"
    output("You win!");
else if game_choice is "Save the princess"
    output("You saved the princess");
else if game_choice is "Go home"
    output("Wimp");
```

7. Write an algorithm for playing a robotic version of the treasure hunt game. The treasure hunt game involves players receiving clues that guide them to other clues, until eventually the last clue leads them to a treasure. For example, at the beginning of the game the players might receive a slip of paper that says go to the big oak tree, at the oak tree they might find another slip of paper that says try swimming, so they go to the swimming pool to look for a third clue, and so forth.

   Robots cant read slips of paper, so the clues for a robot treasure hunt are represented by the colors of tiles. Furthermore, robots arent smart enough to think of (or find) oak trees or swimming pools, so every clue in a robot treasure hunt just requires a robot to move one tile in some direction. After doing so,

the robot examines the color of the tile it is then standing on to figure out where to go next, and so forth.

Here are the specific rules for interpreting tile colors as clues:

- White tile means that the next clue is the tile directly in front of the robot
- Blue tile means that the next clue is the tile to the robots left
- Green tile means that the next clue is the tile to the robots right
- Black tile means that the robot should move two tiles backward, then continue interpreting clues based on its new heading
- Yellow tile is a treasure it marks the end of a part of a treasure hunt

Rules for your algorithm:

- Robot can only move forward, turn left, or turn right.
- The robot can detect the color of the position it is current on.

```
current_color = random("White","Blue","Green","Black","Yellow"); // Start off
    ↪ the game by generating a random color

output("Welcome to the \"Treasure Hunt\" game!");

while current_color is not "Yellow"
    if current_color is "White"
        if input("The next clue is in front of you. Would you like to move one
            ↪  block forward? y/n") is "y"
            current_color = random("White","Blue","Green","Black","Yellow");
        else
            exit;
    if current_color is "Blue"
        if input("The next clue is on your left. Would you like to move there?
            ↪  y/n") is "y"
            current_color = random("White","Blue","Green","Black","Yellow");
        else
            exit;
    if current_color is "Green"
        if input("The next clue is on your right. Would you like to move there
            ↪ ? y/n") is "y"
            current_color = random("White","Blue","Green","Black","Yellow");
        else
            exit;
    if current_color is "Black"
        if input("Uh oh, you need to move two tiles backwards. Would you like
            ↪ to move there? y/n") is "y"
            current_color = random("White","Blue","Green","Black","Yellow");
        else
            exit;
if current_color is "Yellow"
    output("You've found the treasure! The game is now over.");
```