EE 200 Exam 1

Tufts University

9 October 2019

Name: ______________________________

Question	Points
Types	16
Numbers and strings	11
Pointers and arrays	18
Tricks with arrays	5
Echo	6
What happens?	12
Total	68
Bonus	2

Instructions:

1. This examination contains 9 pages, including this page.
2. You have **seventy-five (75) minutes** to complete the examination.
3. Write your answers in this booklet. We scan this into Gradescope, so scratch work on other pieces of paper will not be scanned or counted for credit.

Question 1: Types

(a) [4 pts]

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

struct Point p;
struct Point *pointPointer;
char* description = "a pointy pointer";
```

Given the declarations above, what is the result of each `sizeof` operation below? Assume the compiler is GCC and the computer is a 64-bit x86 system (like your laptop or the homework server).

`sizeof Point`

`sizeof p.x`

`sizeof pointPointer`

`sizeof description`

(b) [8 pts] Given the following declarations:

```
int cinnamon = 255;
float nutmeg = 0.001;
double allspice = 1e3;
char ginger = 'c'; // 'c' is ascii 99
```

What are the *datatypes* and *values* of each of these expressions?

`cinnamon + ginger`

`nutmeg * allspice`

`nutmeg / 2.0`

`cinnamon  / 10`

(c) [4 pts] Given the same declarations, what values are printed by the following statements?

```
printf("%u\n", (unsigned int)ginger);
```

```
printf("%c\n", (char)(ginger + cinnamon));
```

Question 2: Numbers and strings

- [4 pts] Write the 16-bit 2's complement binary representation for -16 and 1025.

- [2 pts] Write the hex representation for 162.

- [2 pts] Write the binary representation for 0xF00D.

- [3 pts] Write the declaration for a mutable string initialized to `"Hello!"`.

Question 3: Pointers and arrays

(a) [6 pts] Given the declarations below:

```
double planets[8];
double mars = 6.4171e23;
```

What are the types of these expressions? If they are invalid, say so.

`planets`

`planets[7]`

`*planets`

`&planets`

`&mars`

`*mars`

(b) [4 pts] Given the declaration `float * const * const jupiter`, which of the following can be legally modified?

`jupiter`   `*jupiter`   `**jupiter`   `&jupiter`

(c) [3 pts] What is a null pointer, and why is such a thing useful?

(d) [5 pts] Write a function `times_table` which accepts a 2-D array of `int` and fills it with a multiplication table based on the X and Y indices. For example, running this on a 4×4 array should give:

```
1  2  3  4
2  4  6  8
3  6  9  12
4  8  12 16
```

Your function should work for a square array of any size greater than 0. The top-left (i.e., first) value should always be 1.

Fill in the function arguments below as necessary.

```
void times_table(                                                    )
{
```

Question 4: Tricks with arrays

(a) [3 pts] Suppose we execute the following code:

```
int muffins[3][4];
int* baguette = (int*)muffins;

for(int i = 0; i < 12; i++){
  baguette[i] = i;
}
```

Sketch how x is laid out in memory. Be sure to label the value stored in each element.

(b) [2 pts] What will be printed when the code below runs?

```
printf("%d\n", muffins[0][1]);
printf("%d\n", muffins[1][0]);
```

Question 5: Echo

[6 pts] Write an implementation of the Linux utility `echo`, which simply prints out whatever was passed to it on the command line. For example:

```
$> echo Hello
Hello
$> echo at least I can say that I tried
at least I can say that I tried
```

```
#include <stdio.h>

int main(int argc, char* argv[])
{
```

Question 6: What happens?

In these code segments, the code may not finish, or may not function as intended. Explain what result is printed out, or what happens when the program tries to run.

(a) [3 pts]

```
#define PEAR 10 + 5

printf("first:␣%d\n", 10 / PEAR)
printf("second:␣%d\n", 5 / PEAR)
```

(b) [3 pts]

```
int grapes[10] = {0};
printf("grape␣10␣is␣%d", grapes[10]);
```

(c) [3 pts]

```
int pineapple[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
for(int i = 0; i < 10; i++){
  if(i = 5){
    printf("Halfway!");
  }
  if(i == 9){
    printf("Almost␣done!");
  }
  printf("%d␣", i);
}
```

(d) [3 pts]

```
char* scramble(char* str)
{
  char result[strlen(str)];
  result[0] = str[1];
  result[1] = str[0]; // Very weak scrambling

  return result;
}

printf("breakfast:␣%s\n", scramble("eggs"));
```

Question 7: Bonus

[1 pt] Write down the names of as many of your EE 200 classmates as you can.

[1 pt] Approximately what is the largest value that can be stored in an `unsigned long long int` (64 bits)? Express your answer in decimal or scientific notation (but not as a power of 2).

.