ECE 551D
Fall 2022
Test 1—Version 2

Name: | | NetID: | |

There are 5 questions, with the point values as shown below. You have 70 minutes with a total of 45 points. Pace yourself accordingly.

This exam must be individual work. You may not collaborate with your fellow students. However, this exam is open notes, so you may use your class notes, which must be handwritten by you.

I certify that the work shown on this exam is my own work, and that I have neither given nor received improper assistance of any form in the completion of this work.

Signature:

#	**Question**	**Points**
1	Concepts	5
2	Reading Code	9
3	Testing	12
4	Algorithm	12
5	Writing Code	7
	Total	45

Question 1 Concepts [5 pts]

For all parts of this question, you *must* blacken the circle of the answer you choose. We can *only* see the region with the circles when grading.

1. Based on what you know about the Seven Steps, which of the following is a consequence of writing code without planning first?

   (a) Not having a plan to refer to when you need to debug, which causes great delays.

   (b) Missing important behaviors of the code that cannot be clearly seen without a well-drawn plan.

   (c) Increased chances of introducing bugs when maintaining or adding functionality to the code.

   (d) Realizing you need to completely revise your implementation to address a test case.

   (e) All of the above are consequences.

2. Assuming a system with 8-bit integers, what would be the result of the following operation using signed integers? Specify the numerical result and whether the operation overflows.

   0x18 + 0xD6

   (a) 0xFE, no overflow

   (b) 0x7E, overflow

   (c) 0xEE, overflow

   (d) 0xEE, no overflow

   (e) None of the above

3. Suppose you want to keep track of patients at a clinic. You might want to track the following attributes:

   - Full name
   - Date of birth
   - Phone number
   - Medications

   Which programming construct is the best choice to represent a patient?

   (a) Enum

   (b) Struct

   (c) Typedef

   (d) String

   (e) None of the above

4. Which component of a compiler integrates the program's source code with any header files it includes?

   (a) Linker

   (b) Assembler

   (c) Compiler

   (d) Preprocessor

   (e) None of the above

5. Which one of the following practices is the *best* way to make your code easier to debug?

   (a) Incremental development

   (b) Writing a tail-recursive implementation

   (c) Deep understanding of syntax

   (d) Using Emacs

   (e) None of the above

Question 2 Reading Code [9 pts]

What is the output when the following C code is executed? (Assume appropriate header files have been included.)

```
1  void g(unsigned x, unsigned y) {
2    if (x < y) {
3      printf("%d\n", x);
4      return;
5    }
6    int a = x % y;
7    x /= y;
8    int b = x % y;
9    x /= y;
10   if (a < b) {
11     printf("%d\n", a);
12     g(x*y + b, y);
13   }
14   else {
15     printf("%d\n", b);
16     g(x*y + a, y);
17   }
18 }

19 int main(void) {
20   g(24063, 10);
21   g(73, 4);
22   return EXIT_SUCCESS;
23 }
```

Your output should be 9 lines long. Please write each line where indicated below:

Output line 1
Output line 2
Output line 3
Output line 4
Output line 5
Output line 6
Output line 7
Output line 8
Output line 9

Question 3 Testing [12 pts]

Suppose the following code is correct to solve a certain problem:

```
1  int f(int x, int y, int z) {
2    int temp = x - y;
3    do {
4      temp = temp + 1;
5      z = z - 2;
6    }  while (temp < z);          //Mistake here: temp <= z
7    if (temp % 3 == 0) {
8      return x + temp;
9    }
10   else {
11     return y * z;
12   }
13 }
```

Imagine that instead of this correct code, a programmer makes a mistake on line 6 and writes `temp <= z` instead of `temp < z`.

Write *one* test case that would detect the above error (*i.e.*, exhibit different behavior than the correct program):

- Input:
- Expected return value:
- Return value if mistake were made:

Next, write *one* additional test case which, when combined with your case above, gives you **statement coverage** on the correct code:

x = y = z =

Finally, write *two* additional test cases, such that your test cases together give you **path coverage** on the correct code:

x = y = z =

x = y = z =

Question 4 Algorithm [12 pts]

The table below shows the result of executing an algorithm with one parameter N, which must be a non-negative integer. For values of N from 0 to 6, the algorithm produces the following sequences of numbers:

N	**Output**
0	3 3
1	4 5 6
2	5 3 2 0
3	6 9 10 12 13
4	7 3 2 0 -1 -3
5	8 13 14 16 17 19 20
6	9 3 2 0 -1 -3 -4 -6

Fill in the blanks below to complete the algorithm used to generate these sequences:

Given a non-negative integer N:

Create a variable x and initialize it to _____ .

Create a variable y and initialize it to _____ .

Count from _____ to _____ (inclusive), and

For each number that you count (call it i),

Print the value of _____ .

If _____ is even, then

Update x to be _____ .

Otherwise,

Update x to be _____ .

Update y to be _____ .

Question 5 Writing Code [7 pts]

Translate your algorithm from the previous question into C code. Make sure to specify the parameter(s):

```
#include <stdio.h>

void num_seq(                    ) {









}
```