Student Name: ________________ NetID: ________________ RUID: ________________

Midterm Exam
Tuesday, March 19, 2024
2:00 pm – 3:20 pm
SERC 111

Rutgers University
01:198:211 Computer Architecture Sections 5-8
Spring 2024
Instructor: Prof. Yipeng Huang

Instructions

- Please turn off all electronic devices you have in possession.
- This exam consists of 10 pages. Check that you have all ten pages right away.
- Write your name, NetID, and RUID student number at top right of this page.
- Write your name on the separate bubble sheet and bubble in your 9-digit RUID number under Student Number, using columns 0 through 8. Ignore the 9th column.

- This exam consists of 46 questions, all are single-selection multiple choice out of 5 choices.
- Mark your answers on the bubble sheet which has room for 50 questions. Use only the first 46. Ignore the remaining 4.
- The problems are not sorted by difficulty. Skip ahead to work on sections you find easier.

- This is a closed book, closed notes exam. No electronic devices are permitted. Accessing any prohibited materials during the exam will lead to a score of 0 on this exam.
- We may check your RUID card when collecting the exam booklet and the bubble sheet.

C Programming

The following set of 7 questions are about the C language syntax, arrays, pointers, managing memory, and potential bugs. You can assume the programs are compiled on a computer like iLab, using the following configuration of the gcc compiler (typical for the programming assignments in this class):

```
gcc -Wall -Werror -fsanitize=address -std=c99 -o midterm midterm.c -lm
```

You are asked what will be printed to the command line. Select the correct answer among five options.

1.

```
#include <stdio.h>

void fun (int ptr) {
    ptr = ptr * ptr;
}

int main() {
  int y = 3;
  int* pointer = &y;
  fun ( *pointer );
  printf("%d\n", y);
  return 0;
}
```

A. 3
B. 9
C. 27
D. ERROR: AddressSanitizer: stack-buffer-overflow
E. ERROR: LeakSanitizer: detected memory leaks

Student Name: ________________ NetID: ________________ RUID: ________________

2.

```
#include <stdlib.h>
#include <stdio.h>

int* function( int* pointer ) {
    pointer = malloc ( sizeof(int) );
    pointer[0] = 1;
    return pointer;
}

int main() {
    int* array = NULL;
    array = function( array );
    printf ( "%d\n", array[0] );
    return 0;
}
```

A. -1
B. 0
C. 1
D. 1, followed by ERROR: LeakSanitizer: detected memory leaks
E. ERROR: AddressSanitizer: heap-buffer-overflow

3.

```
#include <stdlib.h>
#include <stdio.h>

void function( int* pointer ) {
    printf( "%d\n", *(pointer+2) );
}

int main() {
    int* array = calloc( 2, sizeof(int) );
    array[0] = 1;
    array[1] = 0;
    function( array );
    free ( array );
    return 0;
}
```

A. 0
B. 1
C. 3
D. ERROR: AddressSanitizer: heap-buffer-overflow
E. ERROR: AddressSanitizer: attempting double-free

4.

```
#include <stdio.h>

int main() {
    int x[1] = {10};
    printf( "%d\n", (*x+1) );
    return 0;
}
```

A. 9
B. 10
C. 11
D. ERROR: AddressSanitizer: stack-buffer-overflow
E. ERROR: AddressSanitizer: heap-buffer-overflow

Student Name: ________________ NetID: ________________ RUID: ________________

5. Suppose that the 5 by 5 2D array ‘matrix’ has already been initialized and populated with data, what two indices from this array would this code snippet swap.

```
int a = matrix[2][4];
int b = *(*(matrix + 4)+1);
 (*(matrix + 2))[4] = *(matrix[3] + 2);

b = matrix[1][4];
matrix[3][2] = a;
```

A. 2,4 and 3,2
B. 2,4 and 4,1
C. 4,1 and 1,4
D. 4,1 and 3,2
E. Nothing is swapped as this code fails at runtime

6.

```
#include <stdlib.h>
#include <stdio.h>

int quizSwap ( int a, int* b ) {
    int temp = a;
    a = *b;
    *b = temp;
    return *b;
}

int main () {
    int x = 2;
    int y = 3;
    int z = quizSwap ( x, &y );
    printf ( "%d\n", x+y+z );
}
```

A. 5
B. 6
C. 7
D. 8
E. 9

7.

```c
#include <stdio.h>

typedef struct treeNode {
    int data;
    struct treeNode* leftNode;
    struct treeNode* rightNode;
} treeNode;

int main() {
    treeNode node3 = {42, NULL, NULL};
    treeNode node4 = {13, NULL, NULL};
    treeNode node5 = {32, NULL, NULL};
    treeNode node6 = {37, NULL, NULL};
    treeNode node1 = {20, &node3, &node4};
    treeNode node2 = {14, &node5, &node6};
    treeNode root = {10, &node1, &node2};

    treeNode* temp = root.leftNode;
    root.leftNode = (&root)->rightNode;
    (&root)->rightNode = temp;

    int a = root.leftNode->rightNode->data;

    printf("%d\n", a);
    return 0;
}
```

A. ERROR: AddressSanitizer: SEGV on unknown address
B. 13
C. 15
D. 37
E. 42

Integers

The following set of 9 questions are about the data representation of bits, bytes, integers, and operations. You can assume the programs are compiled the same way as the last section. You are asked what will be printed to the command line. Select the correct answer among five options.

8.

```c
#include <stdio.h>

int main() {
    unsigned char number = 127;
    number = ~number;
    printf("%d\n", number);
    return 0;
}
```

A. -128
B. -127
C. 1
D. 127
E. 128

9.

```
#include <stdio.h>

int main() {
    unsigned char number = 21;
    number = number<<4;
    printf("%d\n", number);
    return 0;
}
```

A. 80
B. 84
C. 160
D. 168
E. 336

10.

```
#include <stdio.h>

int main() {
    signed short number = -32768;
    number--;
    printf("%d\n", number);
    return 0;
}
```

A. -32769
B. -32767
C. 0
D. 1
E. 32767

11.

```
#include <stdio.h>

int main() {
    signed char number = -64;
    number = number>>8;
    printf("%d\n", number);
    return 0;
}
```

A. -4
B. -2
C. -1
D. 2
E. 6

12.

```
#include <stdio.h>

int main() {
    unsigned char number = 192;
    number = number>>7;
    printf("%d\n", number);
    return 0;
}
```

A. -4
B. -2
C. 1
D. 3
E. 6

13.

```
#include <stdio.h>

int main() {
    printf("%d\n", 9 || 5);
    return 0;
}
```

A. 1
B. 5
C. 9
D. 12
E. 13

14.

```
#include <stdio.h>

int main() {
    printf("%d\n", 10 ^ 2);
    return 0;
}
```

A. 2
B. 8
C. 10
D. 12
E. 100

<table>
<tr><td>15.</td><td>

```
#include <stdio.h>

int main() {
    printf("%d\n", 0100 + 0x100);
    return 0;
}
```

</td><td>A. 132
B. 200
C. 272
D. 320
E. 356</td></tr>
<tr><td>16.</td><td>

```
#include <stdio.h>

int main() {
    int number = 127;
    number = ~number;
    printf("%d\n", number & 0xFF);
    return 0;
}
```

</td><td>A. -128
B. -127
C. 1
D. 127
E. 128</td></tr>
</table>

Perfect numbers binary representations

A perfect number x is a natural number that is equal to the sum of its divisors except for itself, x.

For example, 6 is a perfect number because the positive divisors of 6 are 1, 2, 3, and 6; the sum of the divisors excluding 6 itself is 1+2+3=6. 28 is also a perfect number because the positive divisors of 28 are 1, 2, 4, 7, 14, and 28; the sum of the divisors excluding 28 itself is 1+2+4+7+14=28. 120 is not a perfect number because the positive divisors of 120 are 1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 30, 40, 60, and 120; the sum of the divisors excluding 120 itself does not equal 120.

There is a deep connection between perfect numbers and a type of prime numbers called the Mersenne primes:

If and only if a number x is a perfect number, then $x = 2^{p-1}(2^p - 1)$, where $2^p - 1$ is a Mersenne prime number. Not all values of p yield $2^p - 1$ to be a prime number, and not all values of p yield $x = 2^{p-1}(2^p - 1)$ to be a perfect number.

The first eight perfect numbers x, along with their corresponding value of p and their Mersenne prime, are listed below.

p	$2^p - 1$	x
2	3	6
3	7	28
5	31	496
7	127	8128
13	8191	33550336
17	131071	8589869056
19	524287	137438691328
31	2147483647	2305843008139952128

17.	What is a valid representation of 28?	A. 11111101 B. 0o11100 C. 0x1C D. 0x47 E. 0x11100
18.	What is the binary representation of 496?	A. 110110000 B. 111100000 C. 111101010 D. 111110000 E. 111111000
19.	What is the hexadecimal representation of 496?	A. 1E0 B. 1F0 C. 1F2 D. 1F6 E. 1F8
20.	What is the binary representation of 8128?	A. 111101000000 B. 111111000000 C. 1111110100000 D. 1111111000000 E. 11111111010000
21.	What is the octal representation of 8128?	A. 0o3740 B. 0o17700 C. 0o77640 D. 0o77600 E. 0o77700
22.	What is the binary representation of 33550336?	A. 1010101010101000000000000 B. 1111111111110000000000000 C. 1111111111111000000000000 D. 1111111111111010101010100 E. 1111111111111100000000000
23.	What is the hexadecimal representation of 33550336?	A. 1555000 B. 1FFF000 C. 1FFF554 D. 1FFF700 E. 1FFF800
24.	What is the hexadecimal representation of 8589869056?	A. 1FF700000 B. 1FFF00000 C. 1FFF70000 D. 1FFFF0000 E. 1FFFF7000

Student Name: ________________ NetID: ________________ RUID: ________________

On the iLab machines, char integers are stored using 1 byte, short integers are stored using 2 bytes, integers are stored using 4 bytes, long integers are stored using 8 bytes.

25.	Can the 7th perfect number, 137438691328, be stored correctly as an unsigned char integer (char), as an unsigned short integer (short), as an unsigned integer (int), and an unsigned long integer (long)? Can or cannot to each:	A. Can as char, can as short, can as int, can as long. B. Cannot as char, can as short, can as int, can as long. C. Cannot as char, cannot as short, can as int, can as long. D. Cannot as char, cannot as short, cannot as int, can as long. E. Cannot as char, cannot as short, cannot as int, cannot as long.
26.	Can the 8th Mersenne prime, 2147483647, be stored correctly as a signed int, an unsigned int, a signed long, and unsigned long? Can or cannot to each:	A. Can as signed int, can as unsigned int, can as signed long, can as unsigned long. B. Cannot as signed int, can as unsigned int, can as signed long, can as unsigned long. C. Cannot as signed int, cannot as unsigned int, can as signed long, can as unsigned long. D. Cannot as signed int, cannot as unsigned int, cannot as signed long, can as unsigned long. E. Cannot as signed int, cannot as unsigned int, cannot as signed long, cannot as unsigned long.

As stated earlier, not all values of natural number p yield $2^p - 1$ to be a prime number. In fact, if p is not prime, then $2^p - 1$ will not be prime. And, not all prime values p will yield $2^p - 1$ to be prime. For example, for $p = 4$, $2^4 - 1 = 15$ is not prime.

27.	In the C header file < limits.h >, the constant INT_MAX stores the largest positive signed integer, UINT_MAX stores the largest unsigned integer, LONG_MAX stores the largest positive signed long integer, and ULONG_MAX stores the largest unsigned long integer. Which of these constants might be prime?	A. None of them can be prime. B. INT_MAX C. UINT_MAX D. LONG_MAX E. More than one of these can be prime.

Student Name: ________________ NetID: ________________ RUID: ________________

The IEEE 754 floating point format specifies 32-bit single precision floating point (float) to have k=8 bits in the exp field. The 64-bit double precision floating point (double) has k=11 bits in the exp field.

28.	Can the 8th perfect number, 2305843008139952128, be stored precisely (without any approximation) as a float and as a double?	A. Float: stored as infinity Double: stored as infinity B. Float: stored as infinity Double: imprecisely stored C. Float: imprecisely stored Double: imprecisely stored D. Float: imprecisely stored Double: precisely stored E. Float: precisely stored Double: precisely stored

Floating point numbers

For the following 15 questions, we will explore the properties of an 8-bit quarter precision floating point format. The 8 bits are used in the encoding as follows:

- The most significant (leftmost) bit encodes s, the sign.
- The next k=2 bits encode the exponent. The exponent bias is $2^{k-1} - 1$
- The remaining bits are the frac bits, which encode the mantissa.

The rules are the same as the IEEE 754 standard for 32-bit and 64-bit floating point numbers.

The following 5 questions ask you to match each floating point numerical value to its binary representation, written as hexadecimal. Select the right representation from a shared set of five options.

29.	0.0	A. `0x00` B. `0x60` C. `0x61` D. `0x80` E. `0xE0`
30.	-0.0	
31.	+inf	
32.	-inf	
33.	NaN, not a number	

The following 5 questions ask you to match each special value in this quarter precision floating point number system to its binary representation, written as hexadecimal. Select the right representation from a shared set of five options.

34.	Largest magnitude representable positive number (that is not infinity)	A. 0x01 B. 0x1F C. 0x20 D. 0x5F E. 0x7F
35.	One	
36.	Smallest magnitude normalized positive number	
37.	Largest magnitude denormalized positive number	
38.	Smallest magnitude positive number (that is not 0.0)	

The following 5 questions ask you to match each real number numerical value to its representation in this quarter precision floating point format, written as hexadecimal. Select the right value from a shared set of five options.

39.	-1.8125	A. 0x1F
40.	0.96875	B. 0x32
41.	pi / 2.0	C. 0x5F
42.	pi + 1.0	D. 0xBA
43.	3.9375	E. 0xE0

The following set of 3 questions are about properties of the floating point numbers. You can assume the programs are compiled the same way as the first two sections. You are asked what will be printed to the command line. Select the correct answer among five options.

44.

```
#include <stdio.h>
#include <math.h>

int main() {
    float x = -1/INFINITY;
    float y = INFINITY;
    printf("%f\n", x * y);
    return 0;
}
```

A. -inf
B. -1.0
C. -0.0
D. NaN
E. inf

45.

```
#include <stdio.h>

int main() {
    float x = -1/0.;
    float y = -0./x;

    printf("%f\n", y - y);
    return 0;
}
```

A. -inf
B. -0.0
C. NaN
D. 0.0
E. inf

46. Which of these options is a false statement regarding denormalized numbers?

A. Handles very small numbers close to zero.
B. Ensures small gaps between numbers near zero.
C. Creates a smooth transition from the smallest normalized value to zero.
D. Offers higher precision than normalized numbers.
E. Ensures consistent spacing between values near zero.

This concludes the exam.