CS211 Quiz1

October 1, 2025

Version 1.1

Name: _____

Instructions: Print your name at the top of each page. Mark your answers directly on the test, with an x on your chosen solutions. Multiple choice questions have one solution each. Multiple choice questions are worth 6% each.

1) What is the output of the following code snippet?

```
printf("3 / 6 = %i\n", 3/6);
```

a. 1
b. 0.5
c. 1.5
d. 0

2) For one byte values in 2's complement, which of the following is **false**:

a. The sum of 1 and the most positive number will be the most negative number.
b. The negative value of the number 00001111 is 11110001.
c. There are two binary representations of the value 0, one negative and the other positive.
d. Summing the positive and negative of the same magnitude will overflow and leave 00000000 as the result.

3) After the following code, what is the value of a?

```
int a = 4;
int *b = &a;
b[0] *= 2;
*b += 8;
```

a. 8
b. 12
c. 16
d. It depends upon the uninitialized memory at b.

4) On a 64-bit machine, the maximum value of an **int** will be

a. 2^{32}
b. LLONG_MAX as defined in limits.h
c. 2^{64} - 1
d. 2^{31} - 1

5) Which of the following statements about argc and argv are **false**?

a. The values of argc and argv are only known at run time.
b. C requires both argc and argv when declaring the main function.
c. argv is a pointer that points to other pointers.
d. argv[0] is the program name itself.

6) What value will be printed by the following code?

```
#include <stdio.h>
int main(int argc, char** argv) {
    printf("%lu\n", sizeof(char*));
    return 0;
}
```

a. It prints the size of a pointer on the current architecture.
b. it prints the size of an int on the current architecture.
c. it prints the size of a char on the current architecture.
d. It will not compile because sizeof cannot be used with char*.

7) In a little endian system, which of the following is **true**?

a. The sign bit is stored in the first and the last bytes.
b. The sign bit of an integer is stored in the first byte.
c. The first byte is the most significant.
d. The first byte is the least significant.

8) Using the code below, if an instance of the nicePtr struct is passed into someFunction, which statement is **true**?

```
typedef struct nicePtr nicePtr;
struct nicePtr {
    unsigned long* memory;
    size_t elements;
};

void someFunction(nicePtr p);
```

a. The values of the array pointed to by memory will be copied automatically.
b. The value of the memory pointer will be copied, but not the elements of the array.
c. The typedef has an error and this will not compile.
d. Structs must be passed as pointers, so this will not compile.

9) Which of the following statements of the stack and heap is **false**?

a. Stack memory and heap memory grow from opposite ends of the memory space.
b. Heap memory is allocated and freed by the user.
c. Stack memory is allocated and freed by the user.
d. Variable length arrays may allocate memory on the stack or the heap.

10) Which of the following statements about recursion is **true**?

a. All recursive functions can be rewritten to be tail recursive.
b. All recursive functions can be rewritten to consume no additional memory with repeated recursive calls.
c. All recursive functions can be rewritten as loops using a stack.
d. Compilers cannot automatically optimize tail recursive functions to avoid repeated function calls.

11) In C, which statements about arrays is **false**?

a. C supports fixed-size arrays whose lengths are known at compile-time.
b. C supports variable-length arrays, with unknown lengths at compile-time but with memory that is still automatically managed.
c. C supports dynamically allocated arrays, where the user requests memory with user-specific sizes.
d. C supports dynamically resized array structs, which automatically resize when their memory is full.

Version 1.1

Name: _____

```
#include <stddef.h>

typedef struct Stack Stack;
struct Stack {
    char *memory;
    size_t memsize;
    size_t index;
};

Stack newStack(size_t size);
void freeStack(Stack s);
void push(Stack* s, char element);
char pop(Stack* s);
size_t len(Stack s);
```

12) Refer to the stack code above. In the spaces marked, add code to read the file's characters into a stack and print them in reverse order. You do not need to fill every blank line. You may add additional lines to the side as needed.

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

int main(int argc, char** argv) {
    // Open datafile for reading.
    FILE* datafile = fopen(argv[1], "r");

    Stack s = newStack(10);

    // Read in the next character as an int so
    // we can check for the EOF value.
    int input = fgetc(datafile);
    while (input != EOF) {


        ------------------------------------------


        ------------------------------------------
    }

    // Finished with the file
    fclose(datafile);

    // Print the input in reverse order


    ------------------------------------------


    ------------------------------------------


    ------------------------------------------
    putchar('\n');
    freeStack(s);
    return 0;
}
```

Use the following stack code to answer the subsequent question.

```
typedef struct Bits Bits;
struct Bits {
    bool a : 1;
    bool b : 1;
    bool c : 1;
    bool d : 1;
    bool e : 1;
    bool f : 1;
    bool g : 1;
    bool h : 1;
};

typedef union LLUBits LLUBits;
union LLUBits {
    // 8 bytes to a long long
    Bits bits[8];
    long long number;
};
```

13) Write a function that accepts a pointer to an LLUBits struct as its argument and prints out every bit of the long long number in LLUBits, from most significant to least. The function does not return a value. You may solve this as you prefer, with or without using the Bits struct.