CS 330 & CS 332
Final Exam Prep

C Programming Questions – Part 1

TRUE/ FALSE

A preprocessor command makes your code compile faster.

A preprocessor command makes your code compile faster.

FALSE

Preprocessor commands have NO BEARING on compilation speed.

What do preprocessor commands do?

What do preprocessor commands do?

A preprocessor is a text substitution tool the compiler uses before performing the actual compilation. There are several commands, but the ones we've used most often are #include and #define.

A compiler converts a high-level language to executable machine code.

A compiler converts a high-level language to executable machine code.

TRUE

A compiler translates source code into machine-language instructions. Our C compilers do this by way of first converting source files into assembly, then bytecode.

A library is a source file that contains ready-made functions.

A library is a source file that contains ready-made functions.

TRUE

This is exactly what a C library is!

C functions cannot call themselves.

C functions cannot call themselves.

FALSE

C functions CAN call themselves!
The language supports recursion.

A struct is a user defined data type.

A struct is a user defined data type.

TRUE

A struct is a user defined data type.

What are the user defined data types in C?

- Structs
- Unions

  a collection of different data types, but only one member can contain a value.
- Typedefs

  creates an alias (new name) for a data type that already exists.
- Enums

  consists of a set of named values.

Converting a variable from an int to a float will never affect its value.

Converting a variable from an int to a float will never affect its value.

FALSE

While both ints and floats are 4 bytes in size, a large enough int would get truncated when converted to a float, because not all its width is used to represent a whole number.

While loops are faster than for loops.

While loops are faster than for loops.

FALSE

While loops are NOT faster than for loops.

A do-while loop will always execute at least once.

A do-while loop will always execute at least once.

TRUE

A do-while loop will execute its statement first, BEFORE checking the loop condition.

C is an object-oriented language.

C is an object-oriented language.

FALSE

While structs allow us to implement some OOP principles in the language, C is NOT object-oriented.

C is a low-level language.

C is a low-level language.

FALSE

While C gets closer to the wire than the languages we may have learned prior, (Python, Java), C is itself a high-level language.

The & and && operators are functionally equivalent

The & and && operators are functionally equivalent

FALSE

& is the bitwise and operator and && is the logical and operator.

The size of a pointer is always 8 bytes.

The size of a pointer is always 8 bytes.

TRUE

Regardless of the data type to which it is pointing, a pointer is always 8 bytes wide.

MULTIPLE CHOICE

Which standard library includes the printf() and scanf() functions.

A. <time.h>
B. <stdlib.h>
C. <stdio.h>
D. <printer.h>

Which standard library includes the printf() and scanf() functions.

A. <time.h>
B. <stdlib.h>
C. <stdio.h> CORRECT
D. <printer.h>

What will the following program print?

```
main {
        float f = 9.45;
        int i = f;
        i += 0.55;
        f = i;
        printf("%f", f);
}
```

A. 9.0
B. 10.0
C. 9.55
D. 1.0

What will the following program print?

```
main {
        float f = 9.45;
        int i = f;
        i += 0.55;
        f = i;
        printf("%f", f);
}
```

A. 9.0 CORRECT
B. 10.0
C. 9.55
D. 1.0

What is the correct format specifier to print characters?

A. %d
B. \c
C. %c
D. %lf

What is the correct format specifier to print characters?

A. %d
B. \c
C. %c CORRECT
D. %lf

What will the following program print?

```
main {
        int i;
        for (i = 0; i < 10; i++)
                i += 2;
        printf("%d", i);
}
```

A. 9
B. 10
C. 11
D. 12

What will the following program print?

```
main {
        int i;
        for (i = 0; i < 10; i++)
                i += 2;
        printf("%d", i);
}
```

A. 9
B. 10
C. 11
D. 12 CORRECT

Between gets() and fgets(), which is the safer function?

A. gets()
B. fgets()
C. They are equally safe
D. They are equally unsafe

Between gets() and fgets(), which is the safer function?

A. gets()
B. fgets() CORRECT
C. They are equally safe
D. They are equally unsafe

What will the following program print?

```
main {
    int a, b;
    a = b = 50;
    b /= 2;
    a *= 2;
    printf("%d", ++a + b--);
}
```

A. 126
B. 125
C. 100
D. error

What will the following program print?

```
main {
        int a, b;
        a = b = 50;
        b /= 2;
        a *= 2;
        printf("%d", ++a + b--);
}
```

A. 126 CORRECT
B. 125
C. 100
D. error

Which method is used to convert an integer to a char/string data type?

A. atoi()
B. itoa()
C. itos()
D. ctoi()

Which method is used to convert an integer to a char/string data type?

A. atoi()
B. itoa() CORRECT
C. itos()
D. ctoi()

What will the following program print?

```
main {
    printf("%d", ((3/4) * 60) + 14);
}
```

A. 59
B. 0
C. 14
D. 45

What will the following program print?

```
main {
    printf("%d", ((3/4) * 60) + 14);
}
```

A. 59
B. 0
C. 14 CORRECT
D. 45

Below is a list of different variables. Which option lists these variables by descending size?*

```
char s[5];
int i;
long l;
struct mystruct {
        char x;
        char y;
        int z;
} STRUCT;
char c = '\n';
```

A. s, STRUCT, i, l, c

B. l, s, STRUCT, i, c

C. STRUCT, l, s, i, c

D. l, STRUCT, s, i, c

*Assume that the compiler is NOT adding padding to align data.

Below is a list of different variables. Which option lists these variables by descending size?*

```
char s[5];
int i;
long l;
struct mystruct {
        char x;
        char y;
        int z;
} STRUCT;
char c = '\n';
```

A. s, STRUCT, i, l, c

B. l, s, STRUCT, i, c

C. STRUCT, l, s, i, c

D. l, STRUCT, s, i, c CORRECT

*Assume that the compiler is NOT adding padding to align data.

If we have some variable int var, &var would give us:

A. The address of var.
B. The data type of var.
C. The size of var (in bytes).
D. The value at the location of var.

If we have some variable int var, &var would give us:

A. The address of var. CORRECT
B. The data type of var.
C. The size of var (in bytes).
D. The value at the location of var.

If we have some variable int var, *var would give us:

A. The address of var.
B. The data type of var.
C. The size of var (in bytes).
D. The value at the location of var.

If we have some variable int var, &var would give us:

A. The address of var.
B. The data type of var.
C. The size of var (in bytes).
D. The value at the location of var. CORRECT

Given the following code, what would the output of ptr2 be?

```
int a = 5;
int *ptr1 = &a;
int **ptr2 = &ptr1;
```

A. 5
B. The address of ptr1
C. The address of a
D. The value of ptr1

Given the following code, what would the output of ptr2 be?

```
int a = 5;
int *ptr1 = &a;
int **ptr2 = &ptr1;
```

A. 5
B. The address of ptr1 CORRECT
C. The address of a
D. The value of ptr1

Thank you for coming!

Please write your blazerid on the whiteboard on your way out.