Midterm Exam 2

Fall 2020 COP3223

Short Answers: 8 points each

Your answers to these questions should each be one paragraph or less.

Reflections

1. Explain what an array is and how it is different from an ordinary variable.

An array is different from a variable because an array can hold multiple values, and a variable can hold a single value. An array is also a pointer when it is declared. Other values are just an int or a double etc, while an array is in fact a pointer to an array of memory. This means arrays are treated differently passing through functions.

2. Explain what a pointer is and how it is different from an ordinary variable.

A pointer points to the memory address of a variable. If you were to output the value of a pointer you would get the memory address of whatever it is pointing to, or junk if it is pointing to nothing. However, if you dereference a pointer with an & sign, you can view the content of the address the pointer is pointing to. An array is also a pointer itself. So passing an array to a function can show the differences between pointers and variables.

3. Explain the relationship between pointers and arrays.

Arrays are pointers. When you declare a double array[256]; you are in fact creating a pointer. array is the name of the pointer. That is why when we pass a array to a function, the function requires double *array. Since an array is a pointer, we use the * in front of the array name because we are saying we are taking in a pointer. The only difference is an array is a fixed address while a pointer can take different addresses.

4. Explain, in brief, the process of writing text to a file.

To print to a file you must first declare a file pointer for the file such as: FILE *ofp; Then we must open the file using ofp = fopen("outputfilename.txt", "w"); The first argument in fopen is the filename and the second is the mode we are using. To write to a file we use write mode. From there, printing to a file is just as easy as printing to the terminal. Instead of printf(); we use fprintf(); fprintf() takes the argument of ofp, the file, and then the content of what you wish to print ex: fprintf(ofp, "Hello World"); This would print Hello World into a file called outputfilename.txt.

Details

5. Explain the difference between `char *s;` and `char s[80];`

char *s is a pointer. char s[80] is an array. Although yes, both are pointers technically, char* is a pointer with a name of s, and char s[80] is an array pointer with a name of s. As well, char * is a pointer that will point to a character/string at some address, char s[80] will have a fixed address. These are very similar, however have small differences as said.

6. Explain why `double` is generally superior to `float`.

Double is generally superior because of its 64 bit memory allocation compared to a float's 32 bit memory allocation. Double will be more precise and provide better accuracy, making it superior to a float.

7. Explain why with `char s[80];` we would use `fgets(s, 79, stdin);` and not `fgets(s, 80, stdin);`

In this situation we use fgets(s, 79, stdin); because we have to allow for the null character at the end of the string. At the end of every char array/string there is a '\0' that must be added. So we subtract 1 to allow for this character to be added.

8. Explain the difference between `scanf()`, `fscanf()` and `sscanf()`, and when to use each.

scanf() is used to scan an input from the terminal. fscanf() is used to scan an input from a file. sscanf() is used to scan the contents of a string. In a situation we may read scanf() if we are taking an input from a terminal. A situation where we may use fscanf() would be if we were to read from a file. sscanf() can often be used after an fgets() because we can fgets(buf, 255, stdin) and then sscanf(buf, "%lf", x) to read a double value from the string and assign it to x.

Problems: 36 points total

*Each of these very short programs has something wrong with it. Explain what, **and explain how to fix it**. Your answers should each be two sentences or less.*

Program 1 (7 points)

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

int main(void) {
    char *s[80];
    int i;

    printf("Enter a number: ");
    fgets(s, 79, stdin);
    i = atoi(s);
    printf("You said %d.\n", i);
    return 0;

}
```

The error in this program is you are trying to assign a char array, as well as assigning a char pointer. You have to do one or the other. So in this case, removing the [80] and leaving s as a pointer will fix this program.

Program 2 (7 points)

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

int main(void) {
    int i;

    scanf("%d", i);
    printf("You said %d.\n", i);
    return 0;
}
```

The error with this program is that scanf format specifies int * or a pointer. So when using scanf you have to use an & symbol before the variable to follow the format of scanf. You must assign i by the address not the variable itself in this case. So; scanf(“%d”, &i);

Program 3 (7 points)

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

int main(void) {
    float x;

    scanf("%d", &x);
    printf("You said %d.\n", x);
    return 0;
}
```

The error with this program is that within both of the scanf statements. You are trying to assign an int to a float. And call an int which is not an int. You must use %f instead of %d if you wish to assign a float. Fix:

```
scanf("%f", &x);
printf("You said %f.\n", x);
```

Program 4 (15 points)

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

int main(void) {
    char s[80];
    float x, y;
    FILE *ofp;

    printf("Enter a number to square: ");
    fgets(s, 79, stdin);
    x = atof(s);
    y = x * x;

    ofp = fopen("math-results.txt", "w");
    fprintf(ofp, "%f squared is %f.\n", x, y);
    return 0;
}
```

This program has a simple error, atof is a part of the stdlib.h library :) Simply, this program must have #include <stdlib.h> at the top and it will run with no issues. Also, probably a good idea to close your file after opening it too. Won’t cause an error, but it is bad practice.