CMSC 21 2ND Long Exam

May 22-23, 2022

I. True or False

1. True
2. False
3. True
4. False
5. False
6. False
7. False
8. True
9. True
10. True

II. Provide the answers to the following:

1. Why is it that the first dimension in an array parameter be left unspecified, but not the other dimensions?

Answers: The reason behind this is that the first dimension, for example in 2D, is the number of rows. Thus, leaving it as unspecified will still get the array divided accordingly to the specification of the other dimension, that is if they are specified. Leaving an array without the other dimensions, or in this case the column, unspecified will result to an incomplete array, hence an error.

```
int array[][3] = {1,2,3,4,5,6} // is equivalent to {{1,2,3}, {4,5,6}}
int array2[2][] = {1,2,3,4,5,6} // unknown; dimensions unspecified
```

2. Function Prototype

```
/*a*/ bool isPalindrome(char *string);
/*b*/ float computeAverage(float arr[]);
/*c*/ void reverseSentence(void);
/*d*/ float squareRoot(int num)
```

3. Find the error and explain

a.

```
int fun(void){
    printf("%s", Inside function fun\n");

    int bored(void){
        printf("%s", Inside function bored\n")
    }
}
```

Error/s and Solution/s:

Firstly, the statement to be printed in the first print function has an unpaired quotation mark, therefore making the following statements a string until another quotation mark is entered.

Second, the function bored(void) should not be within the fun(void) function, since the prior function is still being defined. A function can only be nested within a function if it's being called, not defined. Of course, both functions are assumed to be declared at the beginning of the program.

Lastly, both functions return nothing, so it is better to declare them as void.

Corrected Code Snippet:

The format specifier %s is contained in the quotation mark with the other statements as a string. As

```
void fun(void){
    printf("%s, Inside function fun\n");

    bored();
}

void bored(void){
        printf("%s, Inside function bored\n");
}
```

b.

```
int product(int a, int b){
    int result = a * b;
}
```

Error/s and Solution/s:

Here, the function is defined to be returning a value of type integer yet there is no return statement.

Corrected Code Snippet:

```
int product(int a, int b){
    int result = a * b;
    return result;
}
```

c.

```
void fun(float a);
{
    float a;
    printf("%f", a);
}
```

Error/s and Solution/s:

The fun function was supposed to be defined to return void and accept float-type a. However, it ended with ";" as if the function was just being declared, instead of being defined. The solution of which is to remove the semicolon in the first line and also the statement "float a;"

Corrected Code Snippet:

```
void fun(float a){
    printf("%f", a);
}
```

d.

```
1  void sum(void){
2      printf("%s", "Enter three integers: " )
3      int a, b, c;
4      scanf("%d%d%d", &a, &b, %c);
5      int total = a + b + c;
6      printf("Result is %d", total);
7      return total;
8  }
9
```

Error/s and Solution/s:

The function is defined to return nothing (void) yet there is a return statement to return the integer total.

Also, the print statement in line 2 is invalid; it only needs one string parameter. It lacks a semi-colon, too.

Aside from that, it is best to declare the variables at the beginning of the function, including the variable total as int-type.

Corrected Code Snippet:

If function sum() is to return void, then:

```
1   void sum(void){
2       int a, b, c, total;
3
4       printf("Enter three integers: " );
5       scanf("%d%d%d", &a, &b, %c);
6
7       total = a + b + c;
8
9       printf("Result is %d", total);
10  }
```

4. Provide the answers to each of the following. Assumption: integer numbers are stored in 4 bytes, and the first element of the array is at location 2500 in memory.

```c
/*II. 4*/
#include <stdio.h>
#include <stdlib.h>

#define SIZE 5

int main(){
    // a
    int numbers[SIZE] = {1, 2, 3, 4, 5};
    //b
    int *ptr, i;

    //c
    ptr = numbers;

    //d
    printf("(d) Using pointer/offset notation with the pointer ptr: \n");
    for(i = 0; i < SIZE; i++){
        printf("a[%d] = %d\n", i, *(ptr+i));
    }

    //e
    printf("\n(e) Using pointer/offset notation using the array name as the pointer: \n");
    for(i = 0; i < SIZE; i++){
        printf("a[%d] = %d\n", i, *&numbers[i]);
    }

    //f
    printf("\n");
    printf("(f.1) Element 2 of array numbers (array index): %d\n", numbers[1]);
    printf("(f.2) Element 2 of array numbers (array name as pointer): %d\n", *&numbers[1]);
    printf("(f.3) Element 2 of array numbers (pointer index): %d\n", ptr[1]);
    printf("(f.4) Element 2 of array numbers (pointer notation): %d\n", *(ptr + 1));

    return 0;
}
```

Output:

```
C:\Users\Admin\Documents\PROGRAMS>testfile
(d) Using pointer/offset notation with the pointer ptr:
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5

(e) Using pointer/offset notation using the array name as the pointer:
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5

(f.1) Element 2 of array numbers (array index): 2
(f.2) Element 2 of array numbers (array name as pointer): 2
(f.3) Element 2 of array numbers (pointer index): 2
(f.4) Element 2 of array numbers (pointer notation): 2

C:\Users\Admin\Documents\PROGRAMS>
```

g.

"ptr+2" is a reference to the address of the element 2 of the array number. Since the location of the first element of the array is at 2500 in memory, and each integer consumes 4 bytes, the address of ptr+2 is 4 * 2 bytes from 2500, which is ***2508*** in memory.

The value stored at 2508 is the numbers[2] which is ***3***.

5. Find the error in the codes in a-d given initial code.

```
int *xp; //references array x
void *vp = NULL;
int num;
int x[5] = {1, 2, 3, 4, 5};
vp = arr;
```

a. ++xp;
   This is an error since the pointer is not assigned to a value yet; it is yet to be initialized.
b. Num = xp; //use pointer to access first element (assume xp is initialized)
   An error; the assignment should be num = *xp, to retrieve the value of the first element in the initialized array
c. num = *xp[1]; //assign element 1 (value 2) to num
   An error; the statement should just be “num = xp[1];” Doing so would result to an error

```
C:\Users\Admin\Documents\PROGRAMS>gcc -o testfile2 testfile2.c
testfile2.c: In function 'main':
testfile2.c:48:7: error: invalid type argument of unary '*' (have 'int')
   48 |  num = *xp[1];
      |        ^~~~~
```

d. ++x;
   An error; an array cannot be incremented.

III. Application

Source code for #1 and #2: https://github.com/RedPill-Neo/CMSC21/tree/main/2nd%20Long%20Exam