CMSC 21 2nd Long Exam
May 22-23, 2022

I. True or False
1. When calling a function, if the parenthesis is missing, the function won't get called.
- True

2. The following version of sum_array is illegal: int sum_array(int a[n], int n){ … }
- **False**

3. C allows functions to be nested.
- **False**

4. The function prototype double average(); is illegal.
- **True**

5. Suppose a variable fun is declared as a global variable with a value of 10 and redeclared as a local variable with a value of 5. If fun is printed inside the main function, the output is equal to 10.
- **False**

6. If a variable *point* was declared as a pointer and a *var* was declared as an integer variable, **point = &var* is valid.
- **True**

7. The name of an array always points to the value of the first element of an array.
- **True**

Suppose that a is one-dimensional int array and p is a pointer to int variable. Assuming that the assignment p = a has just been performed, which of the following expressions are illegal because of mismatched types? State whether the expression from 8 to 11 is true or not.

8. p == a[0]
- **False**

9. p == &a[0]
- **True**

10. *p == a[0]
- **True**

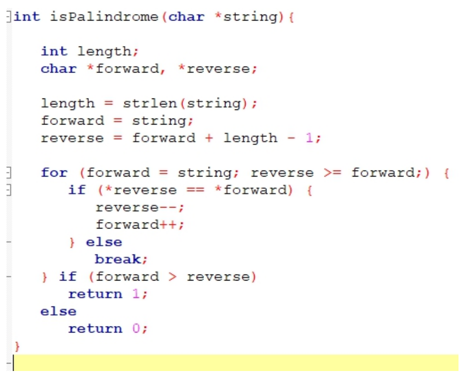
11. p[0] == a[0]
- **True**

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?

- 2d and multidimensional arrays need parameters to be specified because they are both columns and rows involved, which we have to define and give details so it would be more clear and specific to how many rows and columns and how many dimensions does the array contain.

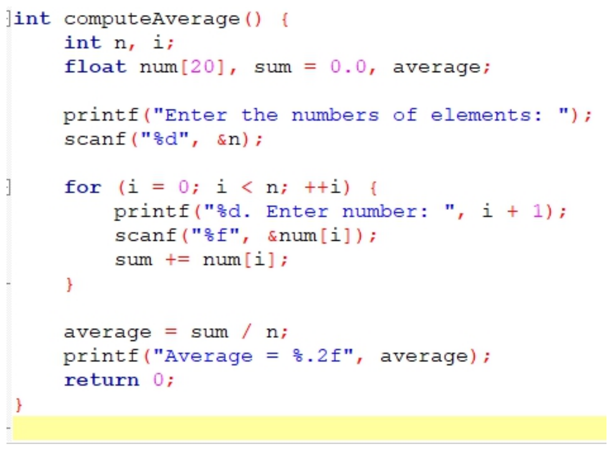
2. Write the function prototype given the following:

a. Function isPalindrome that takes character type pointer argument string and returns a bool value.



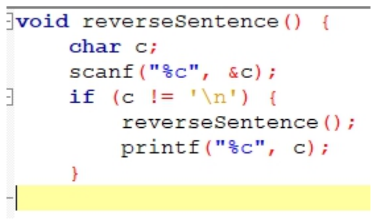
```
int isPalindrome(char *string){

   int length;
   char *forward, *reverse;

   length = strlen(string);
   forward = string;
   reverse = forward + length - 1;

   for (forward = string; reverse >= forward;) {
      if (*reverse == *forward) {
         reverse--;
         forward++;
      } else
         break;
   } if (forward > reverse)
      return 1;
   else
      return 0;
}
```

b. Function computeAverage that takes a floating-point array argument arr (with size of 20) and returns a float value.



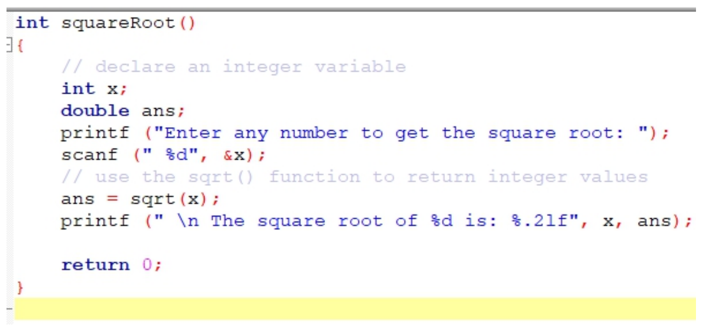
```
int computeAverage() {
    int n, i;
    float num[20], sum = 0.0, average;

    printf("Enter the numbers of elements: ");
    scanf("%d", &n);

    for (i = 0; i < n; ++i) {
        printf("%d. Enter number: ", i + 1);
        scanf("%f", &num[i]);
        sum += num[i];
    }

    average = sum / n;
    printf("Average = %.2f", average);
    return 0;
}
```

c. Function reverseSentence that does not take any argument and returns nothing.



```
void reverseSentence() {
     char c;
     scanf("%c", &c);
     if (c != '\n') {
         reverseSentence();
         printf("%c", c);
     }
```

d. Function squareRoot that takes an integer number num and returns a floating-point result.



```
int squareRoot()
{
    // declare an integer variable
    int x;
    double ans;
    printf ("Enter any number to get the square root: ");
    scanf (" %d", &x);
    // use the sqrt() function to return integer values
    ans = sqrt(x);
    printf (" \n The square root of %d is: %.2lf", x, ans);

    return 0;
}
```

3. Find the error in each of the following code snippets and explain how the error may be Corrected

a. int fun(void){
printf("%s", Inside function fun\n"); **- There's no quotation mark/s before the word inside**
int bored(void){
printf("%s", Inside function bored\n"); **- There's no quotation mark/s before the word inside**
}} **- There shouldn't be function inside another function, separate the other function outside the other**

b. int product (int a, int b){ **- There is no print statement, printf(“%d”, result)**
int result = a * b;
}

c. void fun (float a); **- we can replace void with int instead because the function aims to return a value and omit the float a inside the function**
{
float a;
printf("%f", a);
}

d. void sum(void){ **- We can omit the void as a parameter, or replace the first void with int, and the parameter as (int a, int b, int c)**
printf("%s", "Enter three integers: ") **- There shouldn’t be “%s” before the statement, remove**
int a, b, c; **- Declaration should be at the topmost inside the function and we can put this inside the parameter instea**d
scanf("%d%d%d", &a, &b, &c);
int total = a + b + c;
printf("Result is %d", total);
return total; **- should be return 0**
}

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.

a. Define an integer array numbers with size = 5. Initialize the elements to values 1, 2, 3, 4, 5. Assume a constant SIZE is defined to 5.

```
int arr[5]=[1,2,3,4,5];

SIZE = 5;
```

b. Define an integer pointer, ptr.

```
int *ptr;
```

c. Assign the address of the first element of array numbers to the pointer variable ptr.

```
*ptr == arr[0]
```

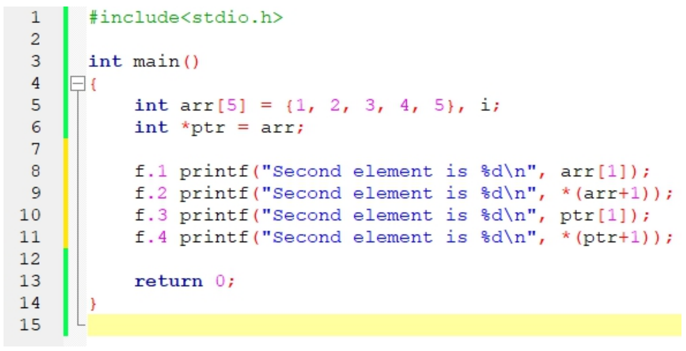
d. Print the elements of array numbers using pointer / offset notation with the pointer ptr

```
#include<stdio.h>

int main(){

    int arr[5] = {1,2,3,4,5};
    int *ptr = &arr;

    for(int i=0; i<5; i++)
        printf("%d ",*(ptr+i));

    return 0;
}
```

e. Print the elements of array numbers using pointer/offset notation using the array name as the pointer

```
#include <stdio.h>
int main() {

    int arr[5] = {1,2,3,4,5};

    for (int i = 0; i < 5; ++i)
        printf("%d\n", *(arr + i));
    return 0;
}
```

f. Refer to element 2 of numbers using a pointer/offset notation using (f.1) array index notation, (f.2) pointer notation with array name as the pointer, (f.3) pointer index notation with ptr, (f.4) pointer notation with ptr.



```
#include<stdio.h>

int main()
{
    int arr[5] = {1, 2, 3, 4, 5}, i;
    int *ptr = arr;

    f.1 printf("Second element is %d\n", arr[1]);
    f.2 printf("Second element is %d\n", *(arr+1));
    f.3 printf("Second element is %d\n", ptr[1]);
    f.4 printf("Second element is %d\n", *(ptr+1));

    return 0;
}

```

F.1. arr[1]
F.2. *(arr + 1)
F.3. ptr[1]
F.4. *(ptr + 1)

g. Assuming that ptr points to the address of the first element, what address is referenced by ptr+2? What value is stored at that address?

- The address will be 2508 for the element references by ptr + 2, and has the value of 3 stored in it, the third element in the array.

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; **- Pointer is initialized, no error**
b. num = xp; //use pointer to access first element (assume xp is initialized) **- *xp should be used because it references array x**
c. num = *xp[1]; //assign element 1 (value 2) to num **- should be num = xp[1], since we are looking for the value in the array**
d. ++x; **- Arrays don't support incrementation**

III. Application

- (Uploaded in GitHub)