Homework Assignment 3

Name: **Asrar Syed**

Due by: **October 24, 2024**

Course Section: **CSC 3320-034**

1.

i) (3 points) Consider the following scanf call:

scanf("%f%d%f", &a, &b, &c);

If the user enters 12.3 45.6 128, what will be the value of a, b, and c. Note that a and c are float-type variables and b is an int variable. Support your answer with appropriate explanation.

“a” is float and will correctly assign 12.3 as the value. “b” is int and will stop reading at the decimal point and will only take 45 as the value. “c” is a float but because the .6 from “46.6” is still in the input buffer the scanf will read this as a correct float number and will assign .6 as the value of “c”.

ii) (3 points) Determine the output of the following fragment. Explain how.

i = 10; j = 5;

printf("%d ", i++ - ++j);

printf("%d %d", i, j);

“i++” is a post-increment (evaluate value then increment after)

“++j” is a pre-increment (increment before then evaluate value)

The first print statement will output the result of “10 - 6 = 4”. The second print statement will output two digits “11” and “6”.

iii) (3 points) Determine the output of the following fragment. Explain how.

i = 7;

j = 3 + --i \* 2;

printf("%d %d", i, j);

The output of the print statement will be two digits “6” and “15”. The “i” variable is decremented within the “j” variable using the pre-increment operators. The “15” is just the output from the “j” expression.

2. (5 points) What output the following fragment will produce? Explain why. How would you re-write the fragment if you wanted the intended output (i.e. "one")?

i=1;

switch (i % 3) {

case 0: printf("zero");

case 1: printf("one");

case 2: printf("two");

}

Output: “onetwo”

Reasoning: 1 mod 3 will return 1 so the switch statement will start at case 1 and print the statement “one” and since there is no break statement then it will continue with case 2 and print the statement “two”.

3. (5 points) Write a single expression whose value is either -1, 0, or + 1, depending on whether i is less than, equal to, or greater than j , respectively with and without using conditional operator.

With conditional operator: int result = (i<j)?-1:(i==j)?0:1;

With conditional operator: int result = (i > j) - (i < j);

4. (5 points) Is the following if statement syntactically correct?

if (n >= 1 <= 10)

printf("n is between 1 and 10\n");

It is not syntactically correct because it will not output the intended result (between 1 and 10), though this code will still compile. “n >= 1” will evaluate first and will return 1 or 0, which will than be compared with “<= 10”.

If so, what does it do when n is equal to 0?

When n is equal to 0, it will evaluate “n >= 1” and return a zero and the result will be compared as

“0 <= 10” which will pass as true and will print the statement. This is a syntatical issue of the if-statements.2

5. (4 points) What does the following code produce? Explain in detail.

for (i = 5, j = i - 1; i > 0, j > 0; --i, j = i - 1)

printf("%4d ", i);

This output the value of i, padded to 4 spaces., i.e., “ 5 4 3 2”, this expression can be broken down into three stages. The initialization “i = 5, j = i - 1”, the conditional “i > 0, j > 0” and the update “--i, j = i - 1”. The noteworthy bits of this for loop is the conditional because when we use a comma the expression evaluates both conditions but only returns the result of the last one.

6. (5 points) What output the following code fragment will produce? Explain in detail.

sum = 0;

for (i = 1; i <= 20; i++) {

if (!(i % 3))

continue; ss

sum += i;

}

printf("%d\n", sum);

The output of the code is an integer value of 147. This code sums up all the numbers from 1 to 20, but skips any number that is divisible by 3. As it loops through each number, it adds only those not divisible by 3 to a running total (sum). In the end, it prints this total, which is 147.

7. (3 points) Rewrite the following loop so that its body is empty. Explain your answer.

for (n = 0; m > 0; n++)

m /= 2;

for (n = 0; m > 0; n++, m /= 2)

;

To make the body of the loop empty, we can place the operation m /= 2 directly in the update part of the for loop. The rewritten loop counts how many times you can divide m by 2 until m becomes 0, while also keeping track of the count in n. The loop's body is empty, meaning it only performs the division and counting without any additional actions.

8. (4 points) The following function is supposed to return true if any element of the array a has the value 0 and false if all elements are nonzero. Sadly, it contains an error. Find the error and show how to fix it:

bool has\_zero(int a[], int n)

{

int i;

for (i = 0; i < n; i++)

if (a[i] == 0)

return true;

else

return false;

}

Explain your answer in detail.

The error in the function lies in the else statement so the current logic returns false as soon as it encounters an element that is non-zero. This means the function will stop checking the remaining elements of the array after just one iteration, which is not the intend logic of the function.

We can fix it by correcting the else statement:

**bool has\_zero(int a[], int n)**

**{**

**int i;**

**for (i = 0; i < n; i++) {**

**if (a[i] == 0)**

**return true;**

**}**

**return false;**

**}**

To correct this to only return false after the loop has checked all elements we need to move the return false statement outside of the if-statement.

9. (5 points) What will be the output of the following program? Modify the swap function so that it can swap without using temp variable.

#include <stdio.h>

void swap(int a, int b);

int main(void)

{

int i = 5, j = 10;

swap(i, j);

printf("i = %d, j = %d\n", i, j);

return 0;

}

void swap(int a, int b)

{

int temp = a;

a = b;

b = temp;

}

Since all we need to do is swap integers one method we can implement is this:

The original swap function did not work because it changed the copies of i and j, not the actual values. So, the output stayed as i = 5, j = 10.

By using pointers, the swap function could directly change i and j. It then swapped them without a temporary variable, giving the correct output: i = 10, j = 5.

**#include <stdio.h>**

**void swap(int \*a, int \*b);**

**int main(void)**

**{**

**int i = 5, j = 10;**

**swap(&i, &j); // Pass the addresses of i and j**

**printf("i = %d, j = %d\n", i, j);**

**return 0;**

**}**

**void swap(int \*a, int \*b)**

**{**

**\*a = \*a + \*b; // Add values and store in a**

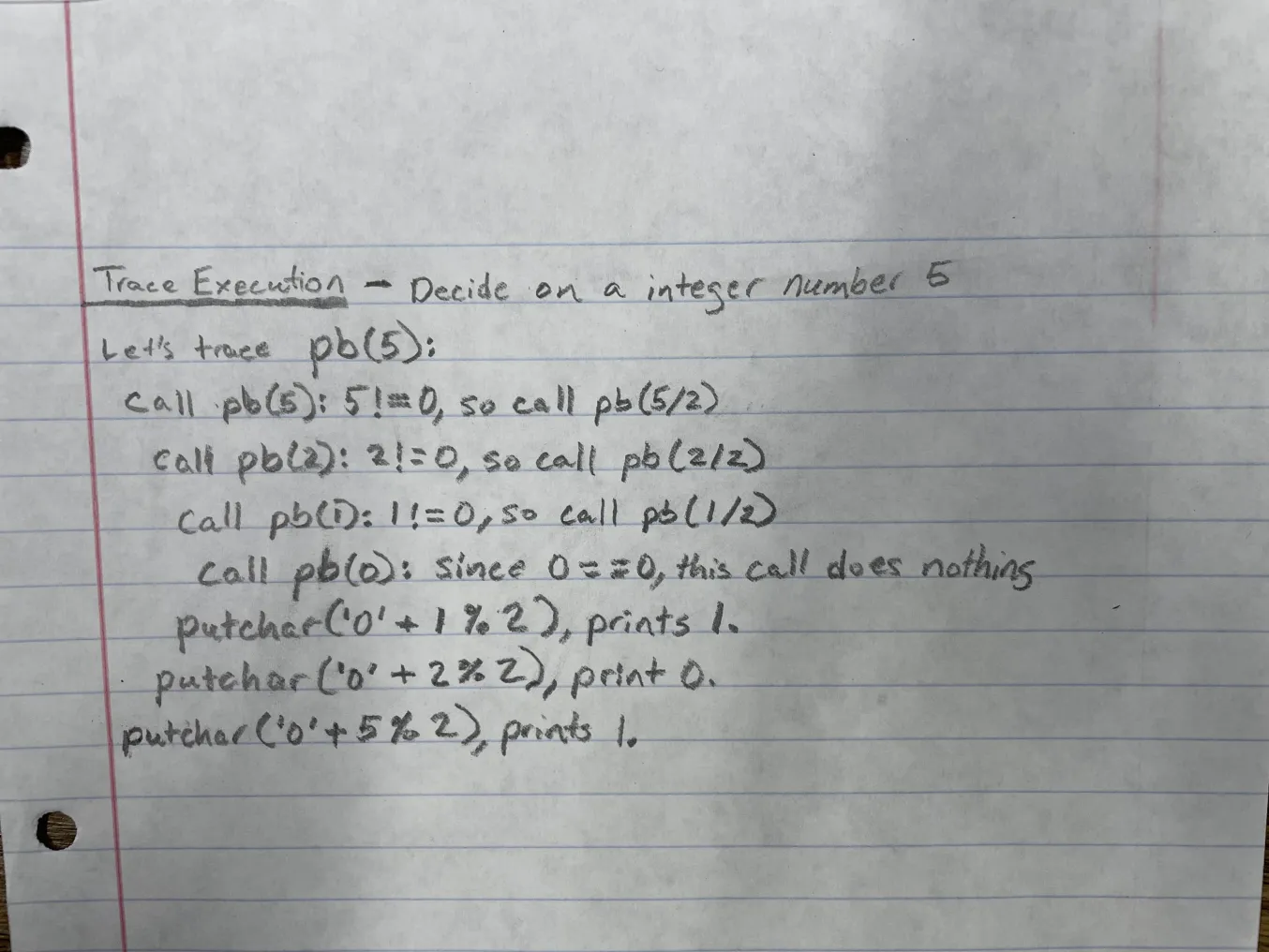
**\*b = \*a - \*b; // Subtract the new b from the sum to get original a**

**\*a = \*a - \*b; // Subtract the new a from the sum to get original b**

**}**

10. (5 points) Trace the execution of the following function by hand (attach screenshots of your trace work). Then write a program that calls the function, passing it a number entered by the user. What does the function do?

This is a recursive function converts an integer to its binary representation and prints it.



void pb(int n)

{

if (n != 0) {

pb(n / 2);

putchar(’0’ + n % 2);

}

}

**int main (void){**

**int number;**

**printf(“Enter a number: ”);**

**scanf(“%d”, &number);**

**pb(number);**

**return 0;**

**}**

11. (2 points (bonus)) The Fibonacci numbers are 0, 1, 1, 2, 3, 5, 8, 13, ..., where each number is the sum of the two preceding numbers. Write a program fragment that declares an array named fib\_numbers of length 40 and fills the array with the first 40 Fibonacci numbers using iterative approach (loops). Hint: Fill in the first two numbers individually, then compute the rest.

i) (3 points (bonus)) Write a recursive function to solve the above problem.

Code is attached on file called fib\_numbers.

You must also submit your .c files. Please check on snowball.cs.gsu.edu if your code executes normally.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Question: | [1](#_bookmark0) | [2](#_bookmark1) | [3](#_bookmark2) | [4](#_bookmark3) | [5](#_bookmark4) | [6](#_bookmark5) | [7](#_bookmark6) | [8](#_bookmark7) | [9](#_bookmark8) | [10](#_bookmark10) | [11](#_bookmark11) | Total |
| Points: | 9 | 5 | 5 | 5 | 4 | 5 | 3 | 4 | 5 | 5 | 0 | 50 |
| Bonus Points: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 5 |
| Score: |  |  |  |  |  |  |  |  |  |  |  |  |