Homework Assignment 4

Name: **Asrar Syed**

Due by: **November 14, 2024**

Course Section: **CSC 3320-034**

1. (5 points) Consider the following program:

#include <stdio.h>

int main( )

{

int arr[4][5];

int i, j;

for (i = 0; i < 4; i++)

{

for (j = 0; j < 5; j++)

{

arr[i][j] = 10 \* i + j;

}

}

printf("%d", \*(arr[1] + 9));

return 0;

}

What is the output of the above program? Explain your answer.

The output is 24. This makes a 2d array called arr of size 4x5 (4 rows and 5 columns). The nested for loops inputs the numbers into the array, where each element in the array (arr[i][j]) is inputed ten times i + j (10 \* i + j). The output is 24, because after the array is populated by the nested for loops it prints the element at arr[2][4].

2. (5 points) What output the following fragment will produce? Explain why.

int fun(char \*str1) {

char \*str2 = str1;

while (\*++str1)

;

return (str1 - str2);

}

int main() {

char \*str = "GeorgiaState";

printf("%d", fun(str));

return 0;

}

The output is **12**. In the main function the pointer \*str variable in the main function is used to hold the starting address of the string "GeorgiaState". The fun function takes the pointer as a parameter and then creates another pointer that also points to the starting address of the string "GeorgiaState". The while loop will iterate over the str1 pointer and increase until it reaches the end of the string (null terminator). Then it will return the length of the string by subtracting the str2 which is pointing at the start of the string with str1 which is at the end. Thus it calculates the total number of characters from start to end, which is 12 for "GeorgiaState".

3. (5 points) An example of a pointer to a function is given below. That is, ptr is pointing to a function fun. What will be the output of the following program?

int fun(int n);

int main() {

// ptr is a pointer to function fun()

int(\*ptr)(int) = fun;

// fun() called using pointer

(\*ptr)(3);

return 0;

}

int fun(int n) {

for (; n > 0; n--)

printf("Computers ");

}

The output is, “**Computers Computers Computers**”. We are passing a value to a function using call by value, in this example, the call **fun(3);** is a call by value. In this method we are passing a copy of the variable and not the actual variable to the called function. When the function fun is executed the for loop runs while **n > 0** and prints “**Computer** ” and **n** decrements by one after each loop.

4. (5 points) Write the following function without using any temp variable:

void swap(int \*p, int \*q);

When passed the addresses of two variables, swap should exchange the values of the variables: swap(&i, &j); /\* exchanges values of i and j \*/

#include <stdio.h>

int main(void)

{

int i = 5, j = 10;

**swap(&i, &j); /\* exchanges values 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

}

5. (5 points) Write the following function:

void find\_two\_largest(int a[], int n, int \*largest, int \*second\_largest);

When passed an array a of length n, the function will search a for its largest and second-largest elements, storing them in the variables pointed to by largest and second\_largest, respectively.

void find\_two\_largest(int a[], int n, int \*largest, int \*second\_largest);

**# initializing largest and second\_largest**

\*largest = a[0];

\*second\_largest = a[0];

for (int i = 1, i < n, i++) {

if (a[i] > \*largest) {

**# update both largest and second\_largest**

\*second\_largest = \*largest;

\*largest = a[i];

}

else if (a[i] > \*second\_largest && a[i] < \*largest) {

**# update only the second\_largest**

\*second\_largest = a[i];

}

}

6. (5 points) Rewrite the following function to use pointer arithmetic instead of array subscripting. (In

other words, eliminate the variables i and j and all uses of the [] operator.) Use a single loop instead of nested loops.

int sum\_two\_dimensional\_array(const int a[][LEN], int n)

{

int i, j, sum = 0;

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

for (j = 0; j < LEN; j++)

sum += a[i][j];

return sum;

}

int sum\_two\_dimensional\_array(const int \*a, int n) {

int sum = 0;

const int \*p = a; // Pointer to iterate through the array

for (p = a; p < a + n \* LEN; p++) {

sum += \*p;

}

return sum;

}

7. (7 points) A palindrome is a string if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. For example, strings like “22/02/2022”, “A man, a plan, a canal – Panama” are palindrome. Alphanumeric characters include letters and numbers.

Given a string s, write a function that returns true if it is a palindrome, or false otherwise. Do not use extra memory space to store the characters of s.

bool isPalindrome(char\* s) {

/\* Implement your code here \*/

}

Complete your program by writing a main function and check if it gives you correct results. For this question, you must also submit your .c files. Please check on snowball.cs.gsu.edu if your code executes normally. **Code File Attached – Called palindrome.c**

8. (5 points) Suppose that we call scanf as follows: scanf("%d%s%d", &i, s, &j);

If the user enters 12abc34 56def78, what will be the values of i, s, and j after the call? (Assume that i and j are int variables and s is an array of characters.) Explain your answer.

scanf stops reading an integer when it encounters a non-digit, so i gets 12.

The %s specifier reads until whitespace, so s gets "abc34".

Finally, j gets the next integer, 56, after whitespace.

**Final values after the call: i = 12, s = "abc34", j = 56**

9. (5 points) What does the following program print? Explain

#include <stdio.h>

int main(void) {

char s[] = "Hsjodi", \*p;

for (p = s; \*p; p++)

--\*p;

puts(s);

return 0;

}

This outputs “**Grinch**”. This program can quickly be broken down into 3 parts, i.e., the initialization of character array **s** with the string **"Hsjodi"**. Then a pointer **p** is set to point at the beginning of **s**. The loop iterates through each character in **s** until it reaches the null terminator **(\0)**. On each iteration, **--\*p** decrements the ASCII value of the character that **p** points to. The last part is where the letters are shifted one letter back in their ASCII value, transforming **"Hsjodi"** into **"Grinch"**.

10. What does the following function do? Explain in detail.

int f(char \*s, char \*t) {

char \*p1, \*p2;

for (p1 = s; \*p1; p1++) {

for (p2 = t; \*p2; p2++) {

if (\*p1 == \*p2)

break;

}

if (\*p2 == ’\0’)

break;

}

return p1 - s;

}

The function goes through each character in **s (using p1)** and checks if it exists in **t (using p2)**.

* If a character in **s** is found in **t**, it moves to the next character in **s**.
* If a character in **s** is not found in **t**, it stops and returns the index of that character in **s**.

The function returns the index of the first character in **s** that is not in **t**.

If all characters in **s** are in **t**, it returns the length of **s**.

1. (3 points) What is the value of f("abcd", "babc")?

The function returns the index of “d”, which is 3.

(b) (3 points (bonus)) What is the value of f("abcd", "bcd")?

The function returns the index of “a”, which is 0.

(c) (2 points (bonus)) In general, what value does f return when passed two strings s and t?

The function returns the index of the first character in **s** that is not found in **t**. If all characters in **s** are found within **t**, it returns the length of **s**.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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: | 5 | 5 | 5 | 5 | 5 | 5 | 7 | 5 | 5 | 5 | 3 | 50 |
| Bonus Points: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 5 |
| Score: |  |  |  |  |  |  |  |  |  |  |  |  |