CSc 3320: Systems Programming

Spring 2021

Homework

# 4: Total points 100

Submission instructions:

1. Create a Google doc for each homework assignment submission. 2. Start your responses from page 2 of the document and copy these instructions on page 1.
3. Fill in your name, campus ID and panther # in the fields provided. If this information is missing in your document TWO POINTS WILL BE DEDUCTED per submission.
4. Keep this page 1 intact on all your submissions. If this *submissions instructions* page is missing in your submission TWO POINTS WILL BE DEDUCTED per submission.
5. Each homework will typically have 2-3 PARTS, where each PART focuses on specific topic(s).
6. Start your responses to each PART on a new page.
7. If you are being asked to write code copy the code into a separate txt file and submit that as well.
8. If you are being asked to test code or run specific commands or scripts, provide the evidence of your outputs through a screenshot and copy the same into the document.
9. Upon completion, download a .PDF version of the document and submit the same.

Full Name: Vivian Do

Campus ID: vdo10

Panther #: 002486640

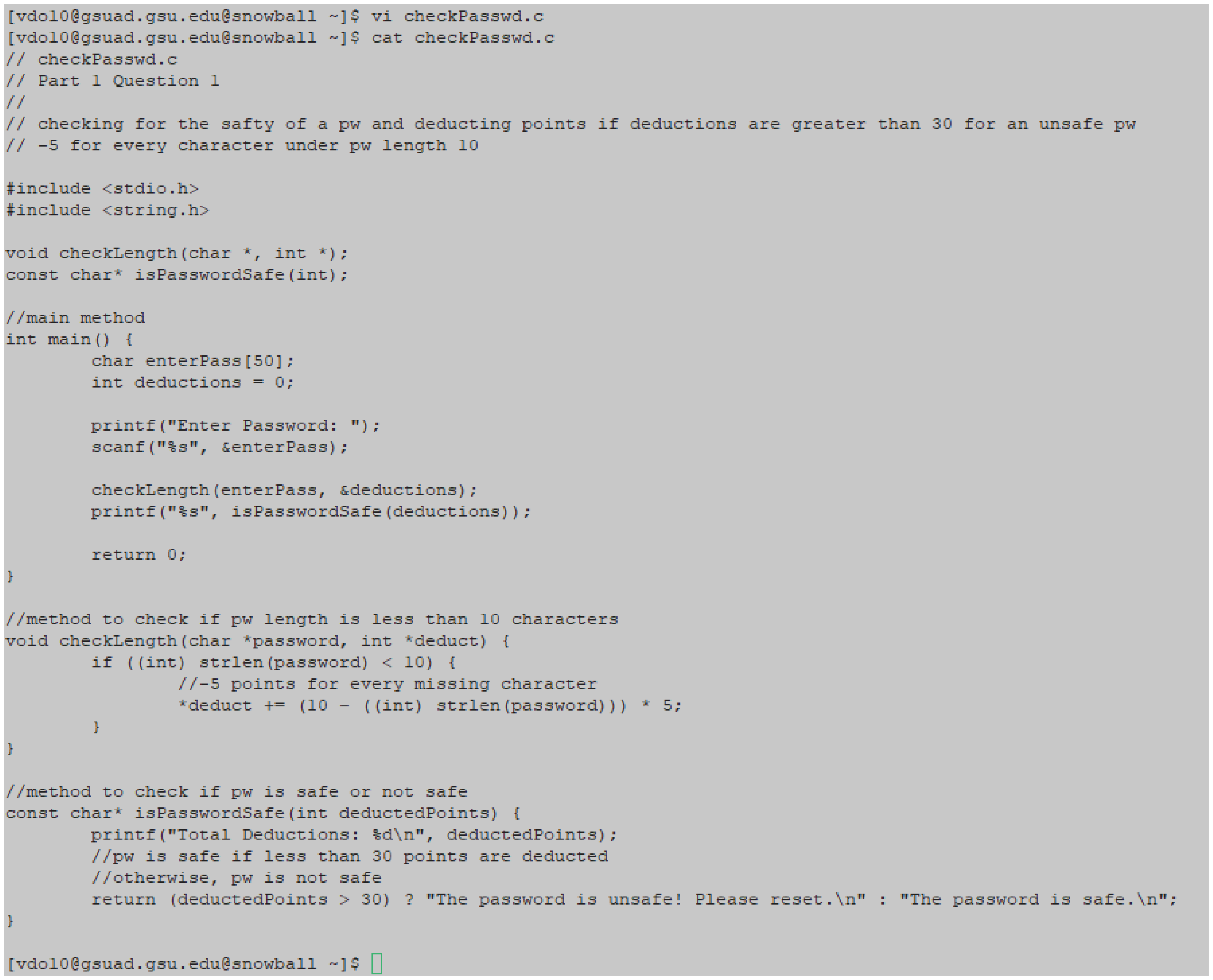
Due Nov 07, 2021

ALL PROGRAMS MUST BE COMMENTED. YOUR SOLUTION WILL NOT BE ACCEPTED IF THERE ARE NO COMMENTS IN YOUR SCRIPT. Also note that the comments MUST be useful and not be random.

PART 1: 40pts
Must incorporate use of Functions and Pointers

1. Write a C program `checkPasswd.c` to check if the length of a given password string is 10 characters or not. If not, deduct 5 points per missing character. If the total deduction is greater than 30 points, print out the deduction and message "The password is unsafe! Please reset."; otherwise, print out "The password is safe."

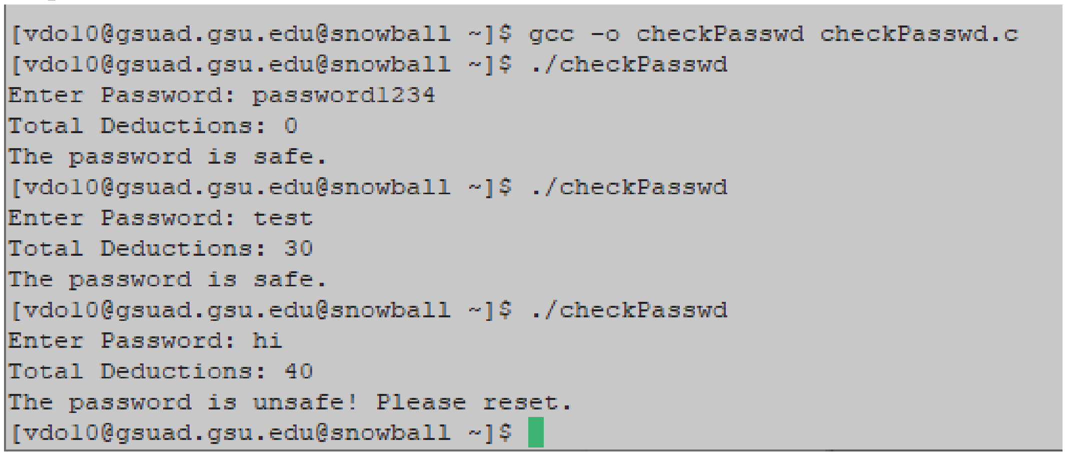
Code:



```
[vdol0@gsuad.gsu.edu@snowball ~]$ vi checkPasswd.c
[vdol0@gsuad.gsu.edu@snowball ~]$ cat checkPasswd.c
// checkPasswd.c
// Part 1 Question 1
//
// checking for the safty of a pw and deducting points if deductions are greater than 30 for an unsafe pw
// -5 for every character under pw length 10

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

void checkLength(char *, int *);
const char* isPasswordSafe(int);

//main method
int main() {
        char enterPass[50];
        int deductions = 0;

        printf("Enter Password: ");
        scanf("%s", &enterPass);

        checkLength(enterPass, &deductions);
        printf("%s", isPasswordSafe(deductions));

        return 0;
}

//method to check if pw length is less than 10 characters
void checkLength(char *password, int *deduct) {
        if ((int) strlen(password) < 10) {
                //-5 points for every missing character
                *deduct += (10 - ((int) strlen(password))) * 5;
        }
}

//method to check if pw is safe or not safe
const char* isPasswordSafe(int deductedPoints) {
        printf("Total Deductions: %d\n", deductedPoints);
        //pw is safe if less than 30 points are deducted
        //otherwise, pw is not safe
        return (deductedPoints > 30) ? "The password is unsafe! Please reset.\n" : "The password is safe.\n";
}

[vdol0@gsuad.gsu.edu@snowball ~]$
```

<u>* a separate code file will be submitted as well for every program in this homework!</u>

Output:

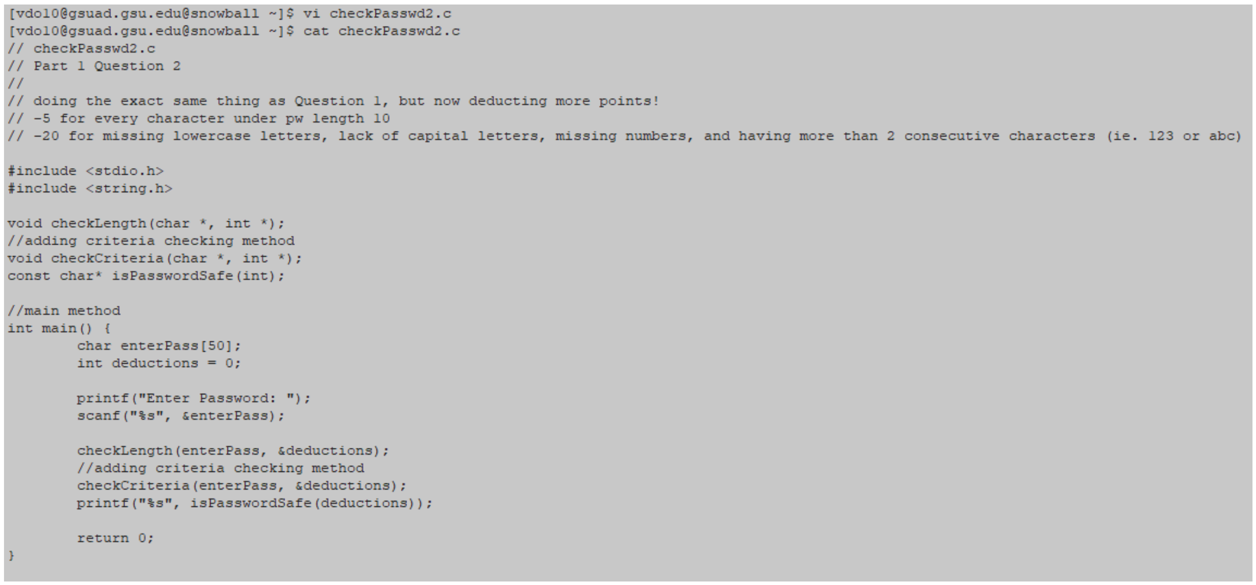


```
[vdol0@gsuad.gsu.edu@snowball ~]$ gcc -o checkPasswd checkPasswd.c
[vdol0@gsuad.gsu.edu@snowball ~]$ ./checkPasswd
Enter Password: password1234
Total Deductions: 0
The password is safe.
[vdol0@gsuad.gsu.edu@snowball ~]$ ./checkPasswd
Enter Password: test
Total Deductions: 30
The password is safe.
[vdol0@gsuad.gsu.edu@snowball ~]$ ./checkPasswd
Enter Password: hi
Total Deductions: 40
The password is unsafe! Please reset.
[vdol0@gsuad.gsu.edu@snowball ~]$
```

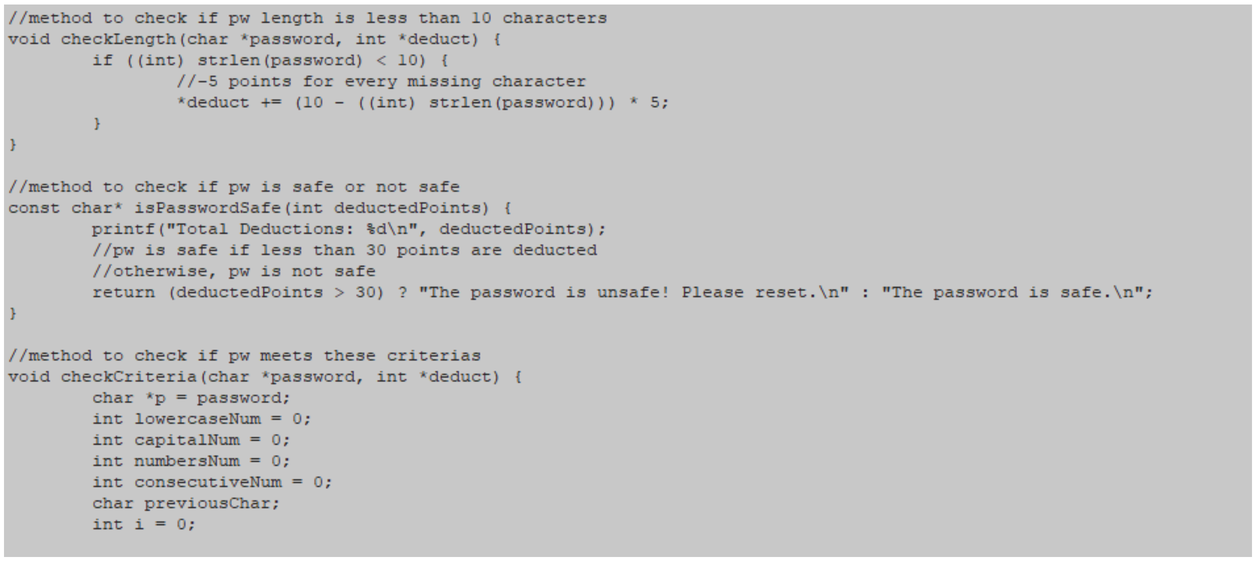
2. Similar to above question, update the C program `checkPasswd.c` to check if a password is safe or by not by checking only the evaluation criteria below. It will still print out the final score, and "safe" or "unsafe" when deduction is more than 30 points.

● Missing lower case -20 points ● Lack of capital letters -20 points
● Missing numbers -20 points ● More than 2 consecutive characters (e.g. 123 or abc) -20 points

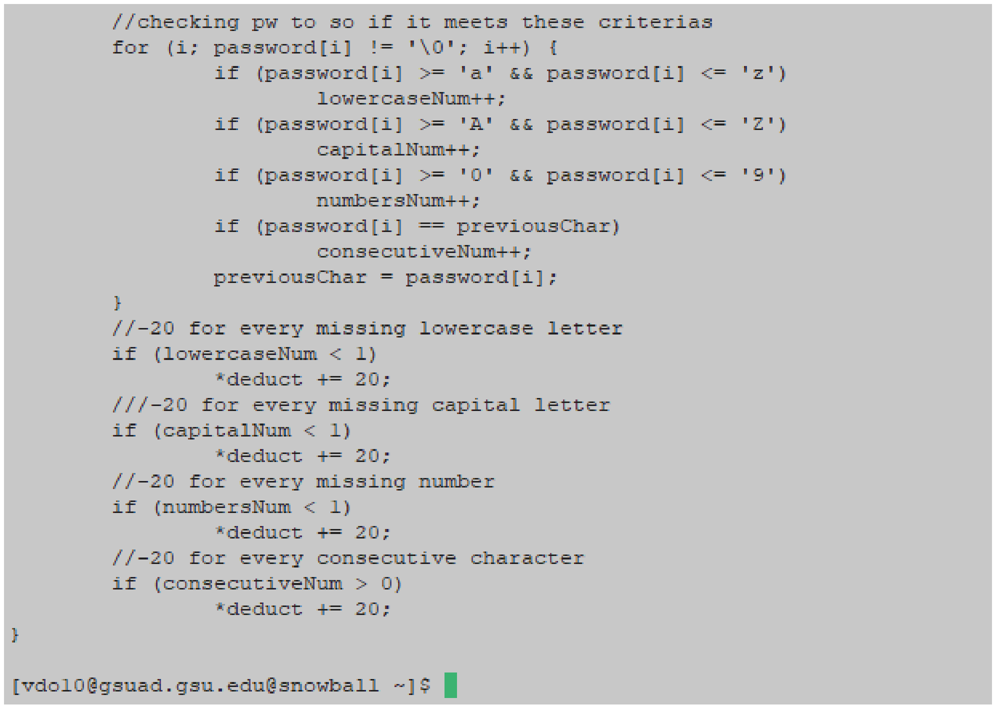
Code:



```
[vdol0@gsuad.gsu.edu@snowball ~]$ vi checkPasswd2.c
[vdol0@gsuad.gsu.edu@snowball ~]$ cat checkPasswd2.c
// checkPasswd2.c
// Part 1 Question 2
//
// doing the exact same thing as Question 1, but now deducting more points!
// -5 for every character under pw length 10
// -20 for missing lowercase letters, lack of capital letters, missing numbers, and having more than 2 consecutive characters (ie. 123 or abc)

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

void checkLength(char *, int *);
//adding criteria checking method
void checkCriteria(char *, int *);
const char* isPasswordSafe(int);

//main method
int main() {
        char enterPass[50];
        int deductions = 0;

        printf("Enter Password: ");
        scanf("%s", &enterPass);

        checkLength(enterPass, &deductions);
        //adding criteria checking method
        checkCriteria(enterPass, &deductions);
        printf("%s", isPasswordSafe(deductions));

        return 0;
}
```

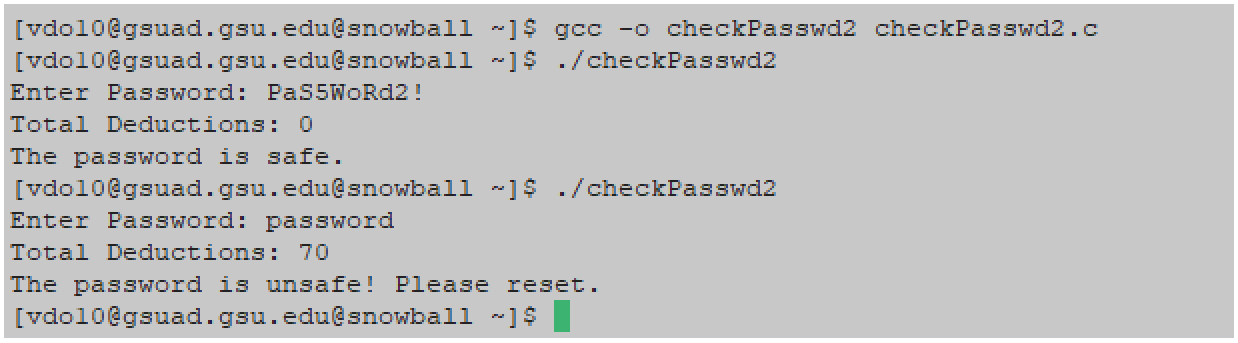


```
//method to check if pw length is less than 10 characters
void checkLength(char *password, int *deduct) {
        if ((int) strlen(password) < 10) {
                //-5 points for every missing character
                *deduct += (10 - ((int) strlen(password))) * 5;
        }
}

//method to check if pw is safe or not safe
const char* isPasswordSafe(int deductedPoints) {
        printf("Total Deductions: %d\n", deductedPoints);
        //pw is safe if less than 30 points are deducted
        //otherwise, pw is not safe
        return (deductedPoints > 30) ? "The password is unsafe! Please reset.\n" : "The password is safe.\n";
}

//method to check if pw meets these criterias
void checkCriteria(char *password, int *deduct) {
        char *p = password;
        int lowercaseNum = 0;
        int capitalNum = 0;
        int numbersNum = 0;
        int consecutiveNum = 0;
        char previousChar;
        int i = 0;
```



```
        //checking pw to so if it meets these criterias
        for (i; password[i] != '\0'; i++) {
                if (password[i] >= 'a' && password[i] <= 'z')
                        lowercaseNum++;
                if (password[i] >= 'A' && password[i] <= 'Z')
                        capitalNum++;
                if (password[i] >= '0' && password[i] <= '9')
                        numbersNum++;
                if (password[i] == previousChar)
                        consecutiveNum++;
                previousChar = password[i];
        }
        //-20 for every missing lowercase letter
        if (lowercaseNum < 1)
                *deduct += 20;
        ///-20 for every missing capital letter
        if (capitalNum < 1)
                *deduct += 20;
        //-20 for every missing number
        if (numbersNum < 1)
                *deduct += 20;
        //-20 for every consecutive character
        if (consecutiveNum > 0)
                *deduct += 20;
}

[vdo10@gsuad.gsu.edu@snowball ~]$
```

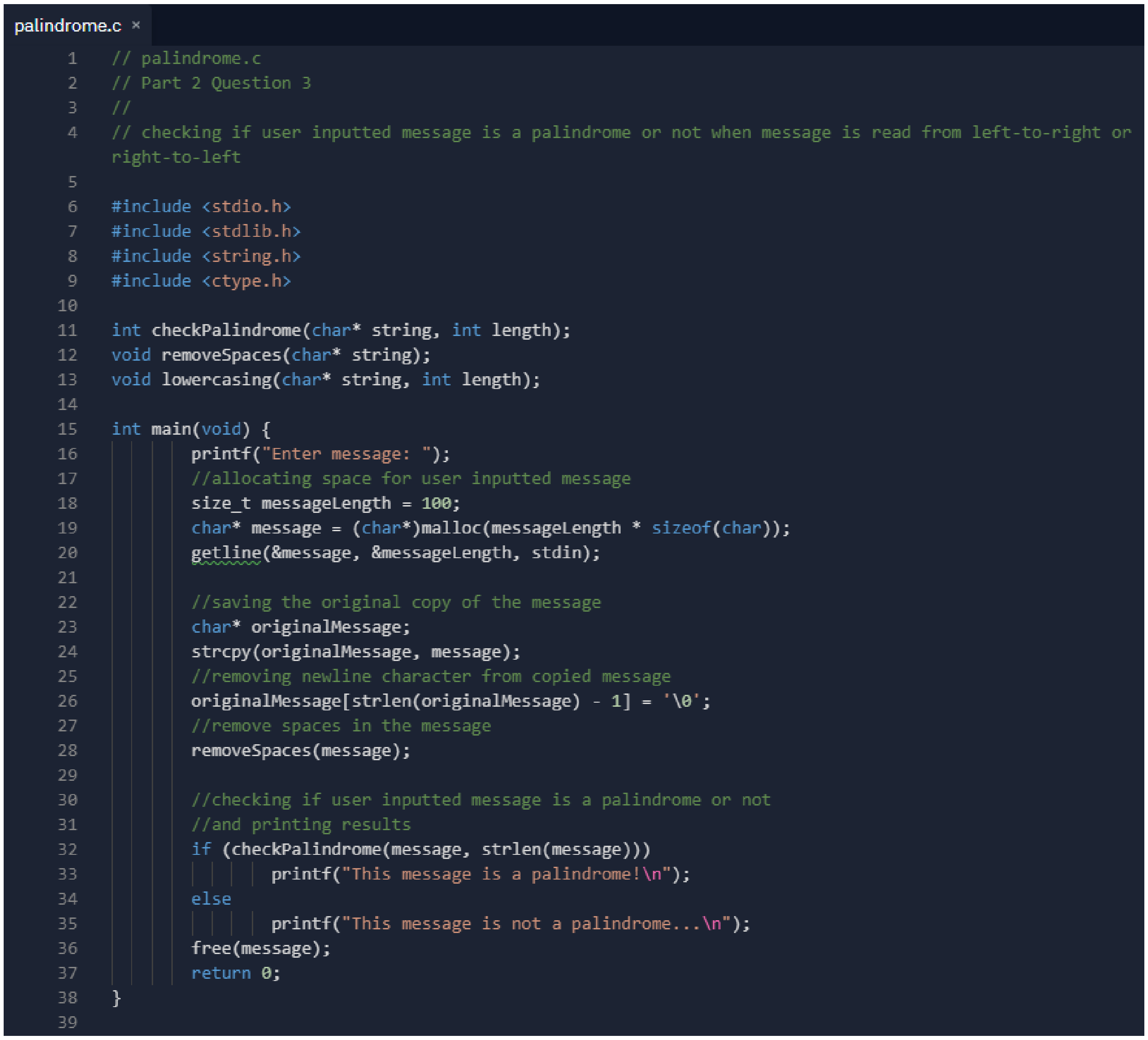
Output:


```
[vdo10@gsuad.gsu.edu@snowball ~]$ gcc -o checkPasswd2 checkPasswd2.c
[vdo10@gsuad.gsu.edu@snowball ~]$ ./checkPasswd2
Enter Password: PaS5WoRd2!
Total Deductions: 0
The password is safe.
[vdo10@gsuad.gsu.edu@snowball ~]$ ./checkPasswd2
Enter Password: password
Total Deductions: 70
The password is unsafe! Please reset.
[vdo10@gsuad.gsu.edu@snowball ~]$
```

Part II : 40pts
Must incorporate the use of Functions and Pointer arrays

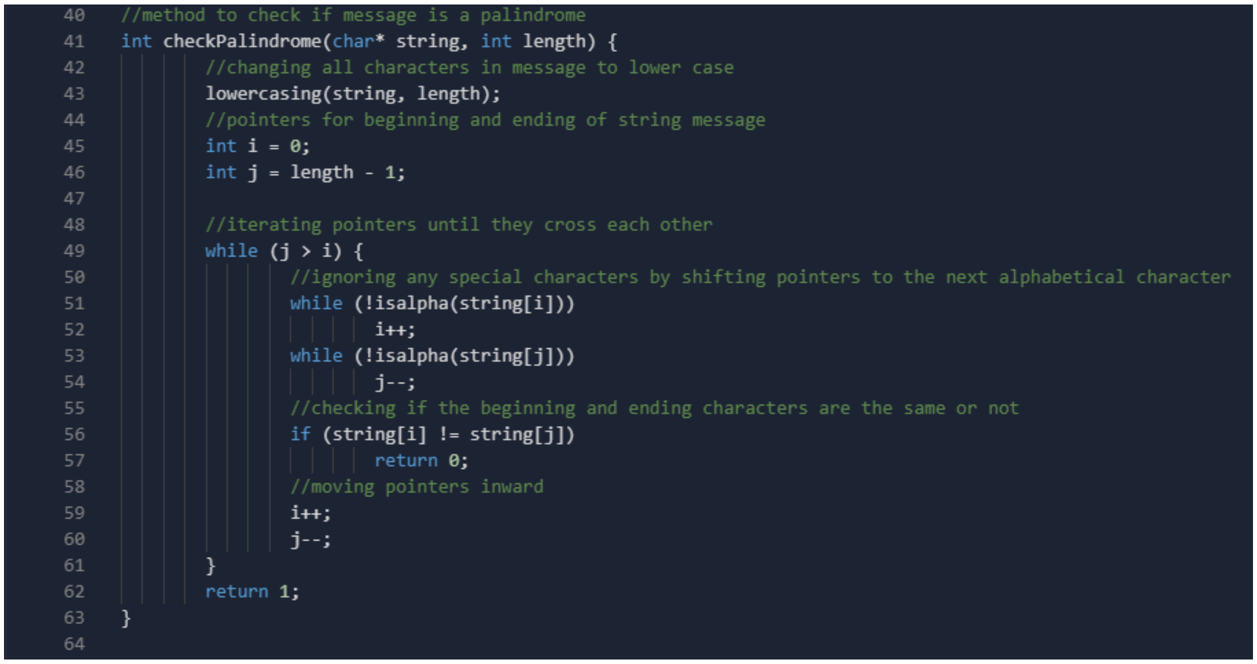
3. Write a program that reads a message (can be characters, numeric or alphanumeric) and checks whether it is a palindrome (the characters in the message are the same when read from left-to-right or right-to-left).

Code:

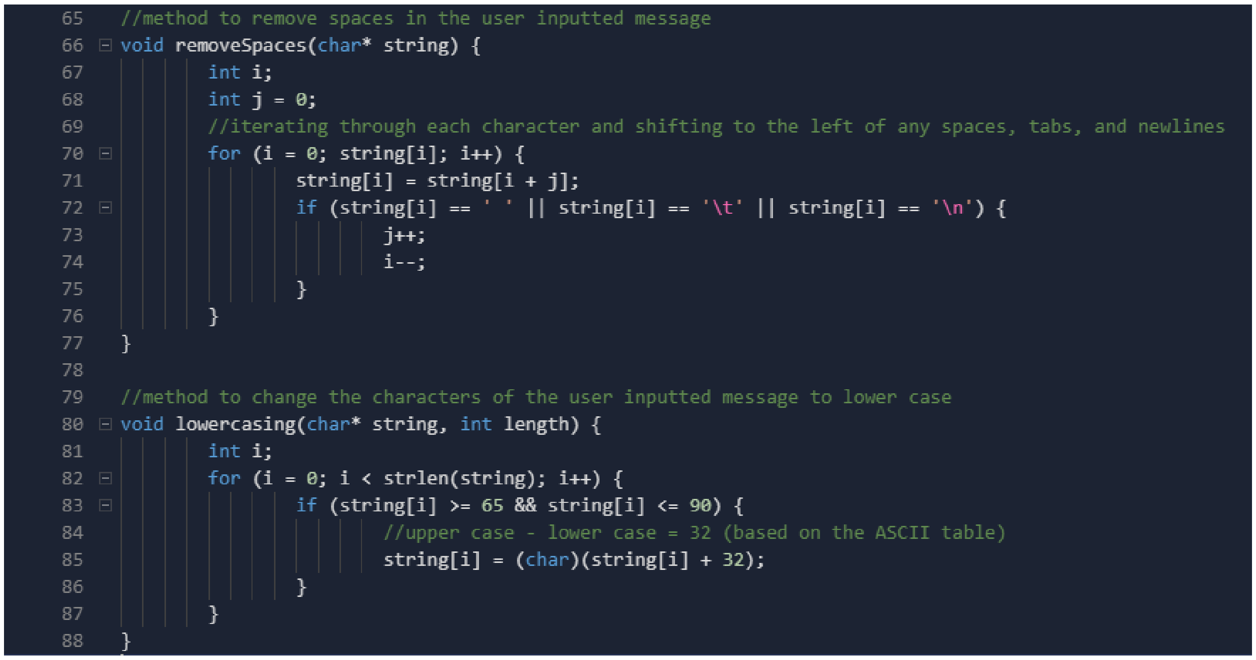


palindrome.c ×

```c
// palindrome.c
// Part 2 Question 3
//
// checking if user inputted message is a palindrome or not when message is read from left-to-right or right-to-left

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

int checkPalindrome(char* string, int length);
void removeSpaces(char* string);
void lowercasing(char* string, int length);

int main(void) {
        printf("Enter message: ");
        //allocating space for user inputted message
        size_t messageLength = 100;
        char* message = (char*)malloc(messageLength * sizeof(char));
        getline(&message, &messageLength, stdin);

        //saving the original copy of the message
        char* originalMessage;
        strcpy(originalMessage, message);
        //removing newline character from copied message
        originalMessage[strlen(originalMessage) - 1] = '\0';
        //remove spaces in the message
        removeSpaces(message);

        //checking if user inputted message is a palindrome or not
        //and printing results
        if (checkPalindrome(message, strlen(message)))
                printf("This message is a palindrome!\n");
        else
                printf("This message is not a palindrome...\n");
        free(message);
        return 0;
}
```

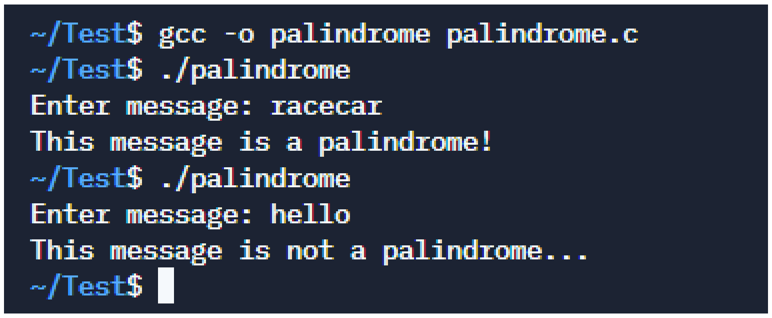


```c
//method to check if message is a palindrome
int checkPalindrome(char* string, int length) {
        //changing all characters in message to lower case
        lowercasing(string, length);
        //pointers for beginning and ending of string message
        int i = 0;
        int j = length - 1;

        //iterating pointers until they cross each other
        while (j > i) {
                //ignoring any special characters by shifting pointers to the next alphabetical character
                while (!isalpha(string[i]))
                        i++;
                while (!isalpha(string[j]))
                        j--;
                //checking if the beginning and ending characters are the same or not
                if (string[i] != string[j])
                        return 0;
                //moving pointers inward
                i++;
                j--;
        }
        return 1;
}

```



```c
//method to remove spaces in the user inputted message
void removeSpaces(char* string) {
        int i;
        int j = 0;
        //iterating through each character and shifting to the left of any spaces, tabs, and newlines
        for (i = 0; string[i]; i++) {
                string[i] = string[i + j];
                if (string[i] == ' ' || string[i] == '\t' || string[i] == '\n') {
                        j++;
                        i--;
                }
        }
}

//method to change the characters of the user inputted message to lower case
void lowercasing(char* string, int length) {
        int i;
        for (i = 0; i < strlen(string); i++) {
                if (string[i] >= 65 && string[i] <= 90) {
                        //upper case - lower case = 32 (based on the ASCII table)
                        string[i] = (char)(string[i] + 32);
                }
        }
}
```

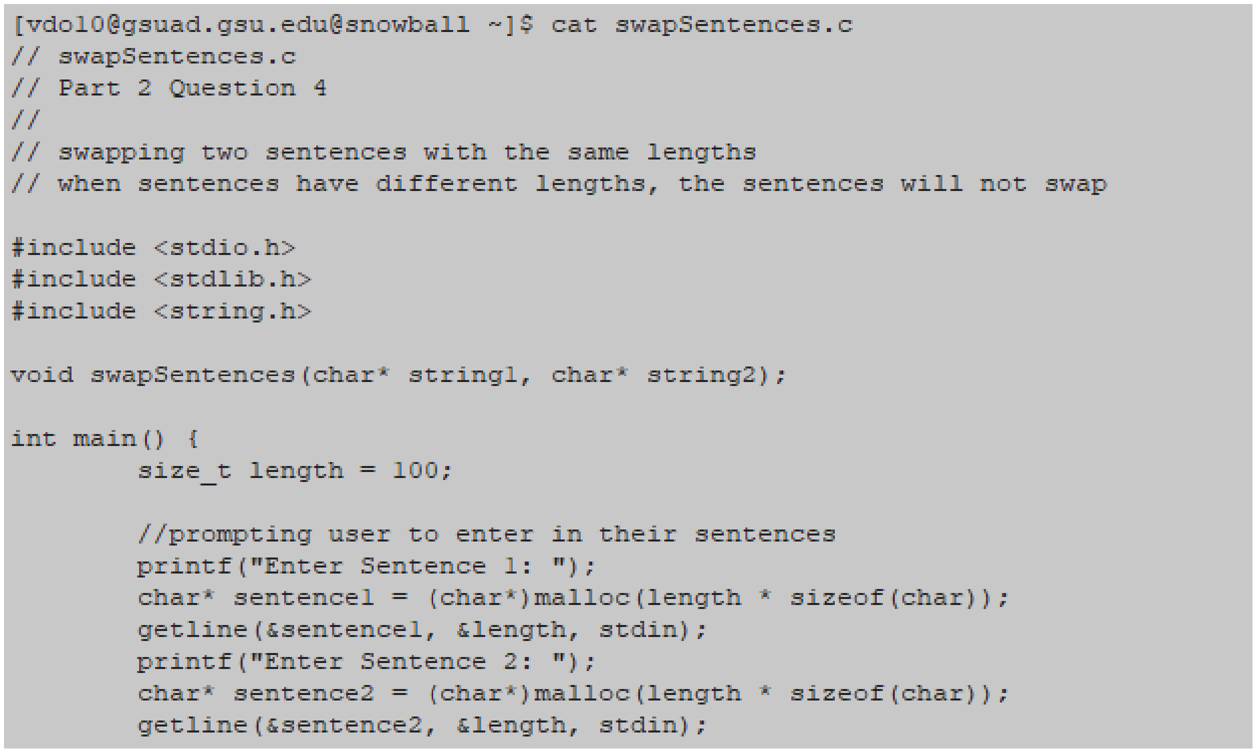
Output:



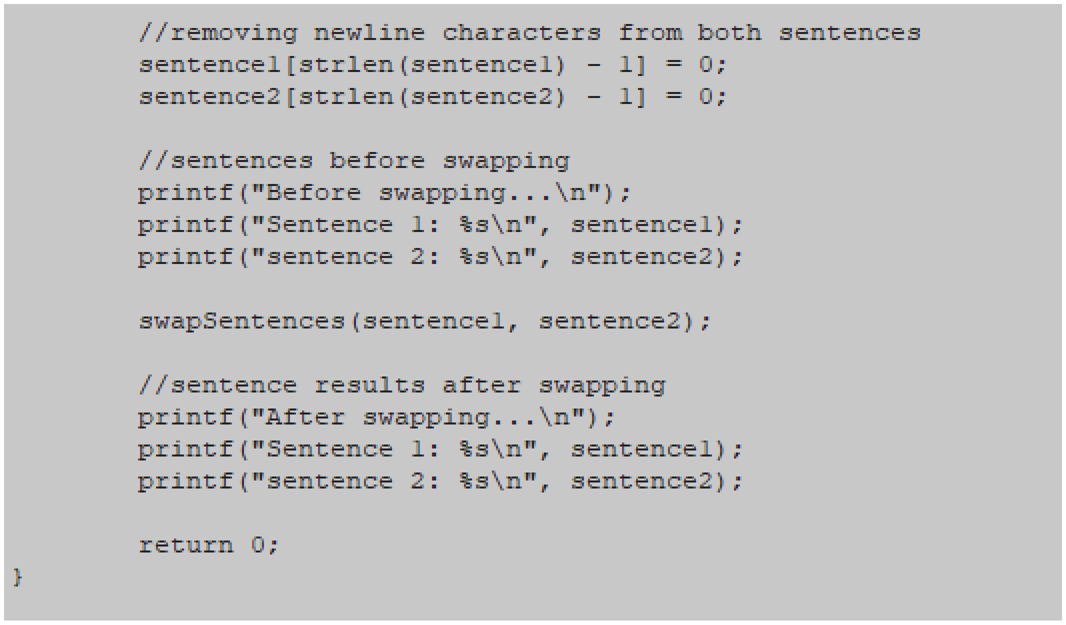
```
~/Test$ gcc -o palindrome palindrome.c
~/Test$ ./palindrome
Enter message: racecar
This message is a palindrome!
~/Test$ ./palindrome
Enter message: hello
This message is not a palindrome...
~/Test$ 
```

4. Write a program that will swap two variables without the use of any third variable. Utilize this program to write a program that reads two sentences that contain alphanumeric characters and the program must swap all the numerics in sentence1 with alphabet characters from sentence 2 and vice-versa. Keep the lengths of the sentences as identical.

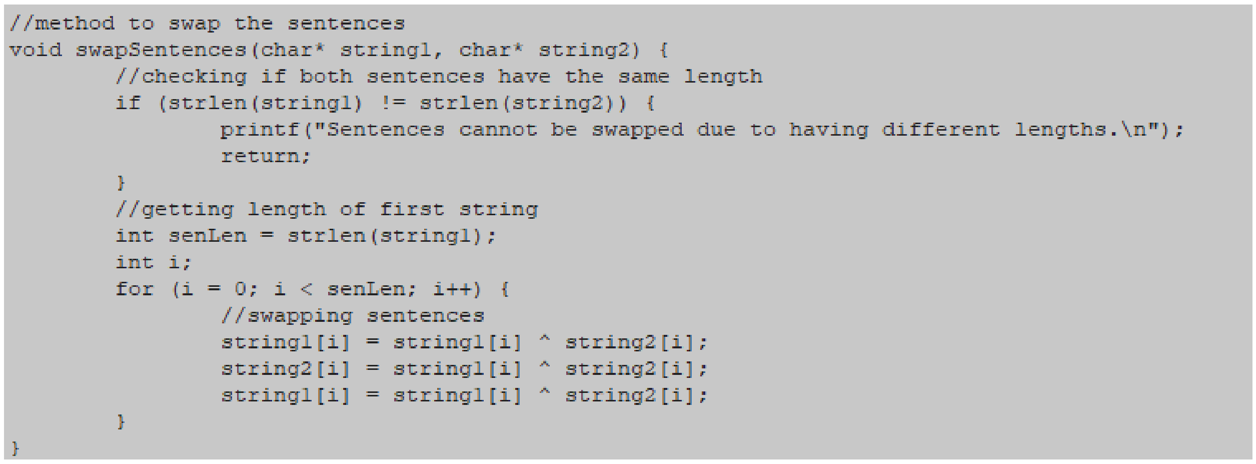
Code:



```
[vdo10@gsuad.gsu.edu@snowball ~]$ cat swapSentences.c
// swapSentences.c
// Part 2 Question 4
//
// swapping two sentences with the same lengths
// when sentences have different lengths, the sentences will not swap

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void swapSentences(char* string1, char* string2);

int main() {
        size_t length = 100;

        //prompting user to enter in their sentences
        printf("Enter Sentence 1: ");
        char* sentence1 = (char*)malloc(length * sizeof(char));
        getline(&sentence1, &length, stdin);
        printf("Enter Sentence 2: ");
        char* sentence2 = (char*)malloc(length * sizeof(char));
        getline(&sentence2, &length, stdin);
```

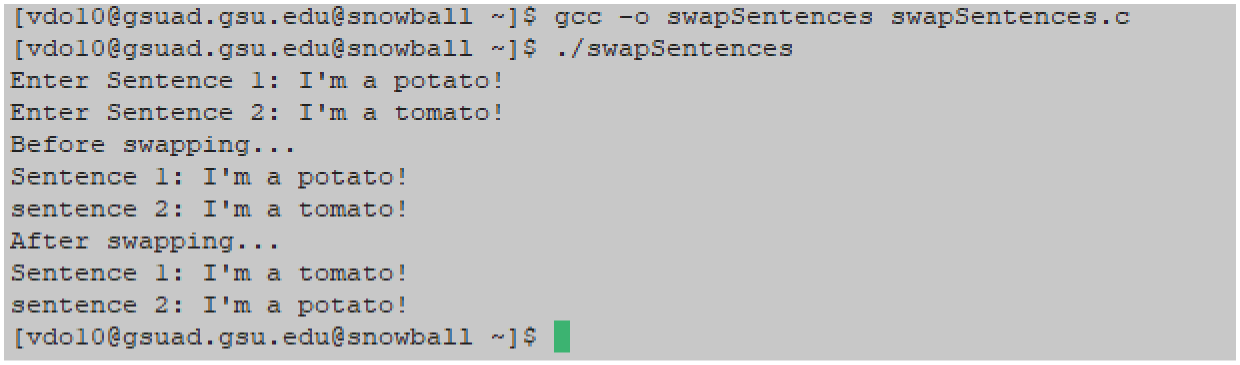


```c
        //removing newline characters from both sentences
        sentence1[strlen(sentence1) - 1] = 0;
        sentence2[strlen(sentence2) - 1] = 0;

        //sentences before swapping
        printf("Before swapping...\n");
        printf("Sentence 1: %s\n", sentence1);
        printf("sentence 2: %s\n", sentence2);

        swapSentences(sentence1, sentence2);

        //sentence results after swapping
        printf("After swapping...\n");
        printf("Sentence 1: %s\n", sentence1);
        printf("sentence 2: %s\n", sentence2);

        return 0;
}
```

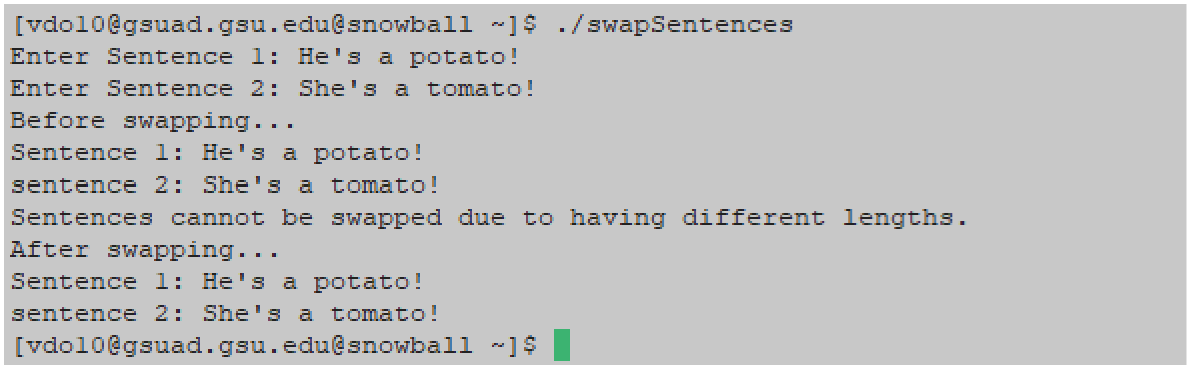


```c
//method to swap the sentences
void swapSentences(char* string1, char* string2) {
        //checking if both sentences have the same length
        if (strlen(string1) != strlen(string2)) {
                printf("Sentences cannot be swapped due to having different lengths.\n");
                return;
        }
        //getting length of first string
        int senLen = strlen(string1);
        int i;
        for (i = 0; i < senLen; i++) {
                //swapping sentences
                string1[i] = string1[i] ^ string2[i];
                string2[i] = string1[i] ^ string2[i];
                string1[i] = string1[i] ^ string2[i];
        }
}
```

Output:



```
[vdo10@gsuad.gsu.edu@snowball ~]$ gcc -o swapSentences swapSentences.c
[vdo10@gsuad.gsu.edu@snowball ~]$ ./swapSentences
Enter Sentence 1: I'm a potato!
Enter Sentence 2: I'm a tomato!
Before swapping...
Sentence 1: I'm a potato!
sentence 2: I'm a tomato!
After swapping...
Sentence 1: I'm a tomato!
sentence 2: I'm a potato!
[vdo10@gsuad.gsu.edu@snowball ~]$
```



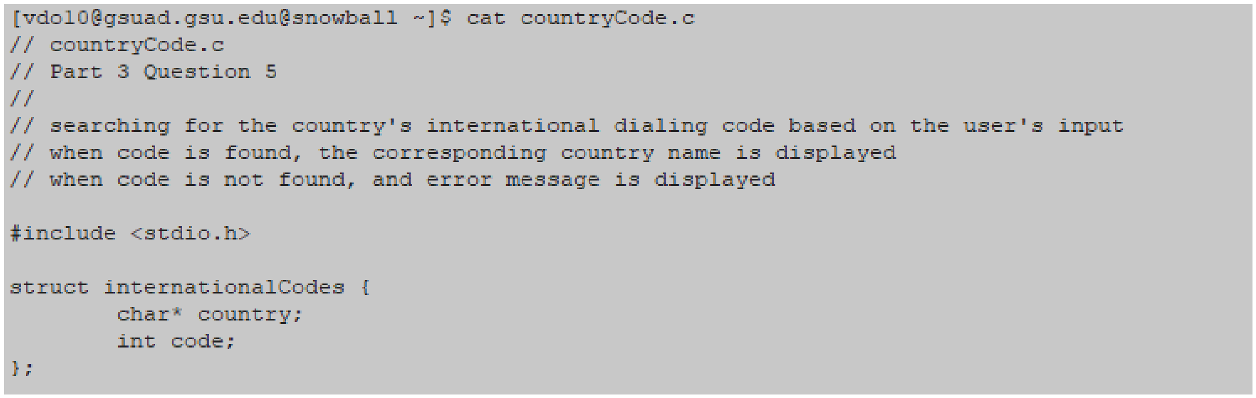
```
[vdo10@gsuad.gsu.edu@snowball ~]$ ./swapSentences
Enter Sentence 1: He's a potato!
Enter Sentence 2: She's a tomato!
Before swapping...
Sentence 1: He's a potato!
sentence 2: She's a tomato!
Sentences cannot be swapped due to having different lengths.
After swapping...
Sentence 1: He's a potato!
sentence 2: She's a tomato!
[vdo10@gsuad.gsu.edu@snowball ~]$ 
```

Part III : 20pts
Must incorporate Functions, Pointers or PointerArrays, and Structures or Unions

5. Write a program that asks the user to enter an international dialing code and then looks it up in the country_codes array (see Sec 16.3 in C textbook). If it finds the code, the program should display the name of the corresponding country; if not, the program should print an error message. For demonstration purposes have at least 20 countries in your list.

   (Programming Project 1 on pg412 in C textbook)

Code:



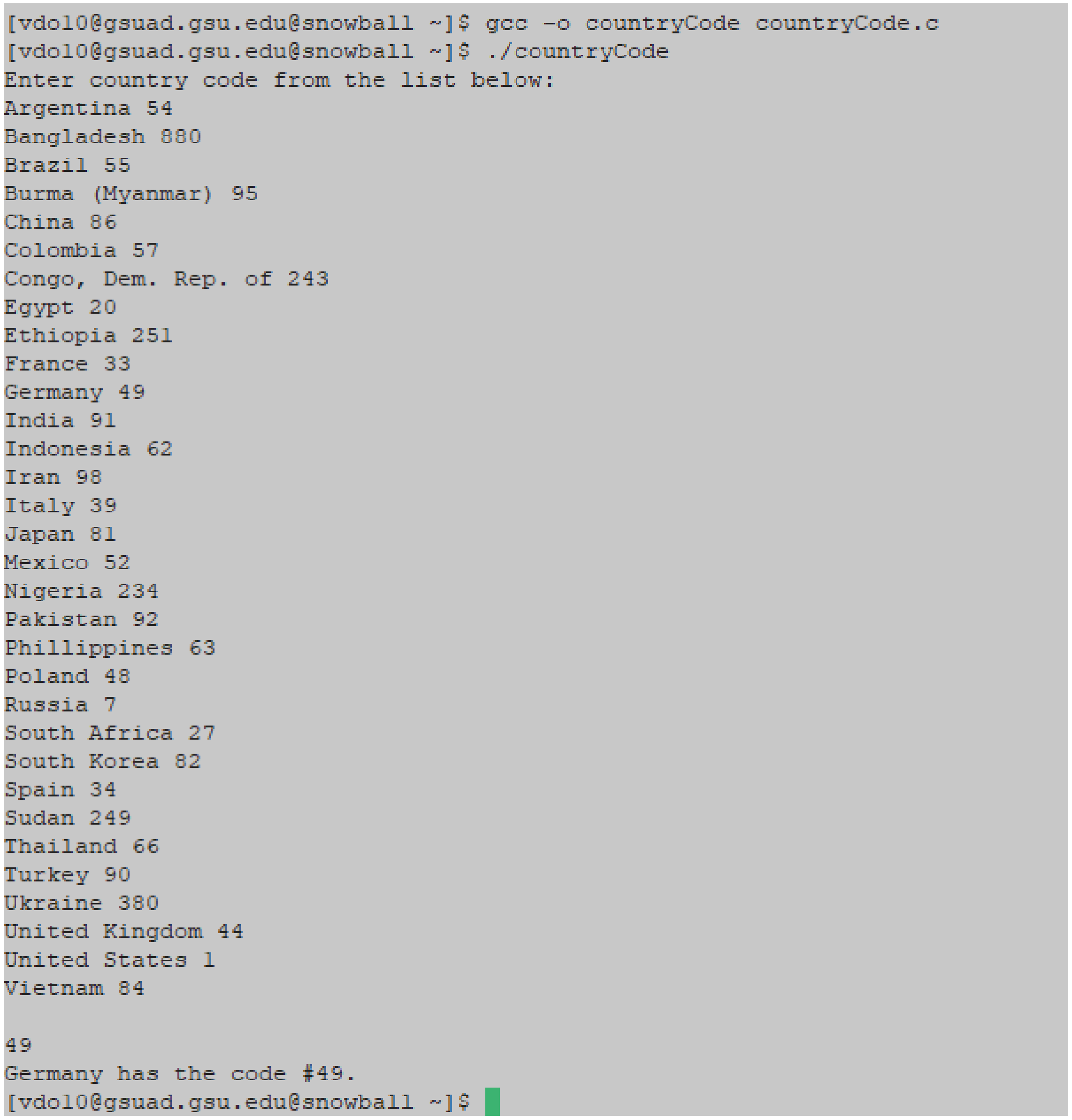
```
[vdo10@gsuad.gsu.edu@snowball ~]$ cat countryCode.c
// countryCode.c
// Part 3 Question 5
//
// searching for the country's international dialing code based on the user's input
// when code is found, the corresponding country name is displayed
// when code is not found, and error message is displayed

#include <stdio.h>

struct internationalCodes {
        char* country;
        int code;
};
```

```c
//creating a struct array with the international dialing codes
const struct internationalCodes countryCodes[] = {
  {"Argentina",           54}, {"Bangladesh",      880},
  {"Brazil",              55}, {"Burma",            95},
  {"China",               86}, {"Colombia",         57},
  {"Dem. Rep. of Congo", 243}, {"Egypt",            20},
  {"Ethiopia",           251}, {"France",           33},
  {"Germany",             49}, {"India",            91},
  {"Indonesia",           62}, {"Iran",             98},
  {"Italy",               39}, {"Japan",            81},
  {"Mexico",              52}, {"Nigeria",         234},
  {"Pakistan",            92}, {"Phillippines",     63},
  {"Poland",              48}, {"Russia",            7},
  {"South Africa",        27}, {"South Korea",      82},
  {"Spain",               34}, {"Sudan",           249},
  {"Thailand",            66}, {"Turkey",           90},
  {"Ukraine",            380}, {"United Kingdom",   44},
  {"United States",        1}, {"Vietnam",          84} };

//displaying all country and country codes
void printCodes();
```

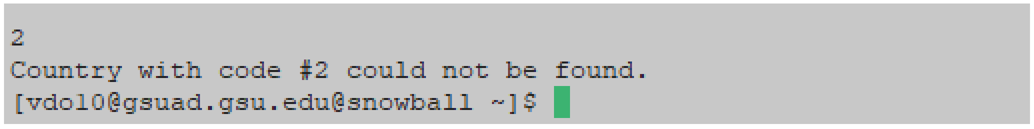
```c
//main method
int main() {
	//finding the total number of elements in array
	int internationalCodeLength = sizeof(countryCodes) / sizeof(countryCodes[0]);
	int code;
	printf("Enter country code from the list below: \n");
	printCodes(internationalCodeLength);
	scanf("%d", &code);
	int codeFound = 0;
	int index = 0;
	int i;
	for (i = 0; i < internationalCodeLength && !codeFound; i++) {
		if (code == countryCodes[i].code) {
			codeFound = 1;
			index = i;
		}
	}
	//checking if code has been found
	if (codeFound) {
		printf("%s has the code #%d.\n", countryCodes[index].country, countryCodes[index].code);
	}
	//checking if code has not been found
	else {
		printf("Country with code #%d could not be found.\n", code);
	}
	return 0;
}
```

```c
//method to display the country and its codes
void printCodes(int length) {
        int i;
        for (i = 0; i < length; i++) {
                printf("%s %d\n", countryCodes[i].country, countryCodes[i].code);
        }
        printf("\n");
}
```

Output:



```
[vdol0@gsuad.gsu.edu@snowball ~]$ gcc -o countryCode countryCode.c
[vdol0@gsuad.gsu.edu@snowball ~]$ ./countryCode
Enter country code from the list below:
Argentina 54
Bangladesh 880
Brazil 55
Burma (Myanmar) 95
China 86
Colombia 57
Congo, Dem. Rep. of 243
Egypt 20
Ethiopia 251
France 33
Germany 49
India 91
Indonesia 62
Iran 98
Italy 39
Japan 81
Mexico 52
Nigeria 234
Pakistan 92
Phillippines 63
Poland 48
Russia 7
South Africa 27
South Korea 82
Spain 34
Sudan 249
Thailand 66
Turkey 90
Ukraine 380
United Kingdom 44
United States 1
Vietnam 84

49
Germany has the code #49.
[vdol0@gsuad.gsu.edu@snowball ~]$
```



```
2
Country with code #2 could not be found.
[vdol0@gsuad.gsu.edu@snowball ~]$
```