Homework: 3

System- Level Programming

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: Cassandra Lundberg

Campus ID: clundberg3

Panther #: 002345582

PART: 1

1) Screenshot of code:

clundberg3@gsuad.gsu.edu@snowball:~

```c
//Name: Cassandra Lundberg   Date: 11/08/21
//Title: checkPasswd.c
//This program checks the length of a given password by input. Then the length
//is compared to 10 or not. For every missing character 5 points will be
//deducted. If the total deduction is greater than 30 then prompt for new
//password


#include <stdio.h>
#include <string.h> //call string class
#define PASS_LEN 10 //define a constant  variable for required password length

main(){ //main method
        char password[PASS_LEN+1]; //create array of characters called password

        //prompt for entering password
        printf("Please enter a password 10 characters long: ");
        //scan input password
        scanf("%s", password);

        //find length of the string password
        int len = strlen(password);

        //while the length of the password is less than 10
        while(len < PASS_LEN){
                        //create variable n to count number of missing characters
                        int n = PASS_LEN - len;
                        //create variable point_deduct
                        int point_deduct = 0;

                        //deduction of points is the missing characters times 5
                        point_deduct = n * 5;
                        //the total final score of points is 100 minus the points deducted
                        int total_point = 100 - point_deduct;

                        //if the points lost are greater than 30
                        if(point_deduct > 30){
                                //print the points deducted, total points, and prompt
                                //to reset the password
                                printf("%d points were deducted\n", point_deduct);
                                printf("%d total points remain\n", total_point);
                                printf("The password is unsafe! Please reset.\n");
                        }
                        else{//otherwise the points deducted are not greater than 30
                                //print the points deducted and the total points
                                printf("%d points were deducted.\n", point_deduct);
                                                                        1,1           Top
```

clundberg3@gsuad.gsu.edu@snowball:~

```
                        printf("The password is unsafe! Please reset.\n");
                }
                else{//otherwise the points deducted are not greater than 30
                        //print the points deducted and the total points
                        printf("%d points were deducted.\n", point_deduct);
                        printf("%d total points remain\n", total_point);
                }

                //prompt to enter another password
                printf("Please enter a password 10 characters long: ");
                scanf("%s", password);
                len = strlen(password);

        }
        //if the length of the password entered is equal to the required length of 10
        if(len >= PASS_LEN ){
                //then print the password is safe
                printf("The password is safe.\n");
        }
        //exit the main method
        return 0;
}
                                                                        63,1          Bot
```

Output of code:

clundberg3@gsuad.gsu.edu@snowball:~

```
[clundberg3@gsuad.gsu.edu@snowball ~]$ vi checkPasswd.c
[clundberg3@gsuad.gsu.edu@snowball ~]$ cc checkPasswd.c
[clundberg3@gsuad.gsu.edu@snowball ~]$ ./a.out
Please enter a password 10 characters long: Cas
35 points were deducted
65 total points remain
The password is unsafe! Please reset.
Please enter a password 10 characters long: Cass
30 points were deducted.
70 total points remain
Please enter a password 10 characters long: Cassandra1
The password is safe.
[clundberg3@gsuad.gsu.edu@snowball ~]$
```

2) Screenshots of code:

clundberg3@gsuad.gsu.edu@snowball:~

```c
//Name: Cassandra Lundberg   Date: 11/08/21
//Title: checkPasswd1.c
//This program checks the length of a given password by input. Then the length
//is compared to 10 or not. For every missing character 5 points will be
//deducted. If the total deduction is greater than 30 then prompt for new
//password. This program also checks for missing lower case letters, lack of capital
//letters, missing numbers, and more than 2 consecutive characters.


#include <stdio.h> //include standard input function
#include <string.h> //include string function liabrary
#include <ctype.h> //include liabrary for testing characters
#include <stdlib.h> //include standard liabrary
#define PASS_LEN 10 //define constant variable called PASS_LEN

//function to check for uppercase and lowercase characters, digits, and consecuative characters with
//input parameters of the input string and the int variable of the points deducted so far
int password_safety(char password[], int point_deduct){
        //create variable i for indexing and variables found_digit, found_upper, and found_lower
        //to flag when found
        int i;
        int found_digit = 0;
        int found_upper = 0;
        int found_lower = 0;
        //for the entire string when an element is not a null character(when the string has not
        //ended)
        for(i = 0; password[i] != '\0'; i++){
                //if an element is a digit then flag the found_digit variable
                if(isdigit(password[i])){
                        found_digit = 1;
                }
                //if an element is an uppercase character then flag the found_upper variable
                if(isupper(password[i])){
                        found_upper = 1;
                }
                //if an element is a lowercase character then flag then found_lower variable
                if(islower(password[i])){
                        found_lower = 1;
                }
        }
        //for every element in the string
        for(i = 0; password[i] != '\0'; i++){
                //if a character is equal to the next character then
                if(password[i] == password[i+1]){
                        //add 20 points to the point_deduct variable
                        point_deduct = point_deduct + 20;
                        //and break out of the loop
```

1,1 Top

clundberg3@gsuad.gsu.edu@snowball:~

```
                //if a character is equal to the next character then
                if(password[i] == password[i+1]){
                        //add 20 points to the point_deduct variable
                        point_deduct = point_deduct + 20;
                        //and break out of the loop
                        break;
                }
        }
        //create variable total to hold the total number of flags not found
        //all the flags are added together and subtracted from 3
        //if a flag is not true then 20 points is added to the variable point_deduct
        //for every flag that is not true add 20 points to the variable
        int total = 3 - (found_digit + found_upper + found_lower);
        point_deduct = point_deduct + (20 * total);
        //return the variable point_deduct with the total number of points to be deducted
        return point_deduct;
}
int main(){ //main function

        //create variables deduct, total_deduct, and final to indicate the number of points deducted
        //and final points remaining
        int deduct = 0;
        int total_deduct = 0;
        int final = 0;

        //while the final points remaining is not 100 then continue prompting for a password reset
        while(final != 100){

                //create string password to hold 20 characters
                char password[20];

                //prompt the user to input a password
                printf("Please enter a password 10 characters long: ");
                //scan input and store in the array of characters called password
                scanf("%s", password);

                //create len to find the length of the string
                int len = strlen(password);

                //if the length of the string is less than 10
                if(len < PASS_LEN ){
                        //ensure the final points remaining is 0 when prompt resets
                        final = 0;
                                //create variable missing to calculate the absolute value
                                //of mssing characters
                                int missing = abs(PASS_LEN - len);
```

89,0-1 46%

clundberg3@gsuad.gsu.edu@snowball:~

```
                                //of mssing characters
                                int missing = abs(PASS_LEN - len);

                                //the variable deduct the number of missing characters times 5
                                deduct = missing * 5;
                                //the total deduct is the return int value of the method
                                //password_safety when the string and current points deducted are
                                //input as arguments
                                total_deduct = password_safety(password, deduct);

                                //final is the total points deducted subtracted from 100
                                final = 100 - total_deduct;

                                //if the total points deducted is less than 30
                                if(total_deduct > 30){
                                        //then print to the user the passweord is not safe
                                        printf("The password is unsafe! Please reset\n");
                                }
                                //tell user the points deducted and the final remaining points
                                printf("%d points were deducted.\n", total_deduct);
                                printf("%d total points remain.\n", final);
                        }
                        else{ //otherwise the length of the string is greater than or equal to 10

                                //the total points deducted current is 0
                                total_deduct = 0;
                                //the total points deducted is now the return in value of the method
                                //password_safety with input arguments the input string and 0
                                total_deduct = password_safety(password, 0);

                                //the final points remaining are the total points deducted subtracted from
                                //100
                                final = 100 - total_deduct;
                                //the points deducted and final points remaining are printed
                                printf("%d points were deducted.\n", total_deduct);
                                printf("%d total points remain.\n", final);

                                //if the total points deducted is greater than 30
                                if(total_deduct > 30){
                                        //than tell the user that the password is unsafe
                                        printf("The password is unsafe! Please reset\n");
                                }
                        }
                }
                //otherwise no points are deducted and the final points remaining are 0
                //the password is safe
                printf("The password is safe.\n");
                                                                                133,1-8        95%
```

clundberg3@gsuad.gsu.edu@snowball:~

```
                                deduct = missing * 5;
                                //the total deduct is the return int value of the method
                                //password_safety when the string and current points deducted are
                                //input as arguments
                                total_deduct = password_safety(password, deduct);

                                //final is the total points deducted subtracted from 100
                                final = 100 - total_deduct;

                                //if the total points deducted is less than 30
                                if(total_deduct > 30){
                                        //then print to the user the passweord is not safe
                                        printf("The password is unsafe! Please reset\n");
                                }
                                //tell user the points deducted and the final remaining points
                                printf("%d points were deducted.\n", total_deduct);
                                printf("%d total points remain.\n", final);
                        }
                        else{ //otherwise the length of the string is greater than or equal to 10

                                //the total points deducted current is 0
                                total_deduct = 0;
                                //the total points deducted is now the return in value of the method
                                //password_safety with input arguments the input string and 0
                                total_deduct = password_safety(password, 0);

                                //the final points remaining are the total points deducted subtracted from
                                //100
                                final = 100 - total_deduct;
                                //the points deducted and final points remaining are printed
                                printf("%d points were deducted.\n", total_deduct);
                                printf("%d total points remain.\n", final);

                                //if the total points deducted is greater than 30
                                if(total_deduct > 30){
                                        //than tell the user that the password is unsafe
                                        printf("The password is unsafe! Please reset\n");
                                }
                        }
                }
                //otherwise no points are deducted and the final points remaining are 0
                //the password is safe
                printf("The password is safe.\n");

                //exit the main method
                return 0;
}
                                                                                137,1          Bot
```

Outputs of code:

clundberg3@gsuad.gsu.edu@snowball:~

```
[clundberg3@gsuad.gsu.edu@snowball ~]$ vi checkPasswd1.c
[clundberg3@gsuad.gsu.edu@snowball ~]$ cc checkPasswd1.c
[clundberg3@gsuad.gsu.edu@snowball ~]$ ./a.out
Please enter a password 10 characters long: cass123
The password is unsafe! Please reset
55 points were deducted.
45 total points remain.
Please enter a password 10 characters long: Ilovemydoggi3
20 points were deducted.
80 total points remain.
Please enter a password 10 characters long: Lundberg1234
0 points were deducted.
100 total points remain.
The password is safe.
[clundberg3@gsuad.gsu.edu@snowball ~]$
```

PART: 2

3) Screenshot of code:

clundberg3@gsuad.gsu.edu@snowball:~

```c
//Name: Cassandra Lundberg  Date: 11/11/21
//Title: palindrome.c
//This program takes a string of numbers or characters or both and checks to see
//if it is a palindrome


#include <stdio.h>
#include <string.h> //included string library function
#define MESS_LEN 100 //created constant variable of 100

void palindrome(char message[], int length){ //palindrome method
        //create variables i for indexing
        int i;
        //create temp string to store the reversed string
        char temp[length+1];
        //create variable j to index the ending element of the string
        int j = length-1;
        //for every element in the string
        for(i = 0; i < length; i++){
                //starting from last element copy all characters from the message
                //string to the temp string
                temp[i] = message[j];
                //decremrent j to move towards the begginging of the string
                j--;
        }
        //compare the two strings useing the strcmp function and if they are equal
        if(strcmp(temp, message) == 0){
                //then the string is a palindrome
                printf("This is a palindrome.\n");
        }
        //otherwise if they are not equal
        else{
                //then the string is not a palindrome
                printf("This is not a palindrome.\n");
        }

}

main(){ //main function
        //create string to store up to 100 elements
        char message[MESS_LEN+1];

        //print prompt to enter message
        printf("Enter a message: ");
        //scan input string
        scanf("%s", message);
                                                                        1,1           Top
```

clundberg3@gsuad.gsu.edu@snowball:~

```c
main(){ //main function
        //create string to store up to 100 elements
        char message[MESS_LEN+1];

        //print prompt to enter message
        printf("Enter a message: ");
        //scan input string
        scanf("%s", message);

        //create variable to store the length of the stirng
        int length = strlen(message);

        //if the length of the string is less than 100
        if(length <= MESS_LEN){
                //then call the method
                palindrome(message, length);
        }
        //otherwise the message is too long and the array cannot hold the entire string
        else{
                //print the message is too long
                printf("That message is too long.");
        }
        //exit the main method
        return 0;
}
                                                                    63,1          Bot
```

Output of code:

clundberg3@gsuad.gsu.edu@snowball:~

```
[clundberg3@gsuad.gsu.edu@snowball ~]$ cc palindrome.c
[clundberg3@gsuad.gsu.edu@snowball ~]$ ./a.out
Enter a message: radar
This is a palindrome.
[clundberg3@gsuad.gsu.edu@snowball ~]$ ./a.out
Enter a message: cassie
This is not a palindrome.
[clundberg3@gsuad.gsu.edu@snowball ~]$ ./a.out
Enter a message: cassic
This is not a palindrome.
[clundberg3@gsuad.gsu.edu@snowball ~]$ ./a.out
Enter a message: caac
This is a palindrome.
[clundberg3@gsuad.gsu.edu@snowball ~]$
```

4) Screenshot of code:

clundberg3@gsuad.gsu.edu@snowball:~

```c
//Name: Cassandra Lundberg  Date: 11/12/21
//Title: swapalphanumeric.c
//This program takes inputs from two different strings of the same length
//containing alphanumeric characters and swaps the numbers of one string
//with the letters fro the other without the use of a temp variable


#include <stdio.h> //library function to incorporate standard inputs
#include <string.h> //library function to incorporate the string function
#define STRING_LEN 100 //constant variable to represent max elements of the strings
void swap(char *p, char *s){ //swap function with inputs of two pointers
        //pointer variable p points to the concatenation of the pointer holding the address of the fi
rst
        //string and the pointer holding the address of the second string
        *p = *p + *s;
        //the pointer variable s points to the address of the first string after the pointer with the
 address
        //of the second string is subtracted from the concatenation of the pointers from above
        *s = *p - *s;
        //the pointer variable p points to the address of the second string after the pointer with th
e
        //address of the first string(now pointer s) is subtracted from the concatenation of the poin
ter
        //from the previous p pointer variable
        *p = *p - *s;
//In doing this the addresses in which the pointers held are now switched
}
int main(void){ //the main function
        //string1 is created with room for 100 elements
        char string1[STRING_LEN+1];
        //string2 is created with room for 100 elements
        char string2[STRING_LEN+1];

        //print prompt for the first sentence
        printf("Enter the first sentence.\n");
        //scan input for string1
        scanf("%s", string1);
        //print prompt for the second sentence
        printf("Enter the second sentence.\n");
        //scan input for string2
        scanf("%s", string2);

        //len1 and len2 are created from the library function of string.h to ensure the strings' leng
th
        //match
        int len1 = strlen(string1);
        int len2 = strlen(string2);
```

"swapalphanumeric.c" 74L, 2877C 1,1 Top

clundberg3@gsuad.gsu.edu@snowball:~

```
        int len2 = strlen(string2);

        //variable i is created for indexing
        int i = 0;
        //while either string has not reached a null character(when either string has not reached the
 end)
        while(string1[i] != '\0' && string2[i] != '\0'){
                //call the swap function with the string's addresses as input for each index
                swap(&string1[i], &string2[i]);
                //then increment the index
                i++;
        }
        //if the lengths match
        if(len1 == len2){
                //then print the prompt for the new first string
                printf("The first sentence is now: ");
                //output the value of string1
                puts(string1);
                //print new line
                printf("\n");
                //print the prompt for the new second string
                printf("The second sentence is now: ");
                //output the value of string2
                puts(string2);
                //print new line
                printf("\n");
        }
        else{//otherwise one of the sentences is larger than the array allows
                //tell user the sentence is too large
                printf("One of the strings is longer than the other.\n");
        }
//exit the main function
return 0;
}
                                                                                    74,1          Bot
```

Output of code:

clundberg3@gsuad.gsu.edu@snowball:~

```
[clundberg3@gsuad.gsu.edu@snowball ~]$ vi swapalphanumeric.c
[clundberg3@gsuad.gsu.edu@snowball ~]$ vi swapalphanumeric.c
[clundberg3@gsuad.gsu.edu@snowball ~]$ cc swapalphanumeric.c
[clundberg3@gsuad.gsu.edu@snowball ~]$ ./a.out
Enter the first sentence.
Cass1379
Enter the second sentence.
John1994
The first sentence is now: John1994

The second sentence is now: Cass1379

[clundberg3@gsuad.gsu.edu@snowball ~]$
```

PART: 3

5) Screenshot of code:

clundberg3@gsuad.gsu.edu@snowball:~

```c
//Name: Cassandra Lundberg  Date: 11/14/21
//Title: international_dial_code.c
//This program takes a dialing code int vlue as input then searches for the int
//value in an array country_codes. If the value is found then the corresponding
//country's name is printed.


#include <stdio.h>  //include the standard input library
#include <string.h> //include the stringn liabrary

struct dialing_code{  //create a structure called dialing_code
        //one member of the structure is a pointer variable called country
        char *country;
        //the other member of structure is an int variable called code
        int code;
};

main(){ //main method
        //create int variable number which will store the input value
        int number;

        //the contant array of strucures called country_codes which has the pointer country pointing
        //to a literal string
        const struct dialing_code country_codes[] =
        {{"Argentina", 54},
        {"Brazil", 55},
        {"China", 86},
        {"Congo, Dem. Rep. of", 243},
        {"Ethiopia", 251},
        {"Germany", 49},
        {"Bangladesh", 880},
        {"Colombia", 57},
        {"Egypt", 20},
        {"France", 33},
        {"India", 91},
        {"Indonesia", 62},
        {"Italy", 39},
        {"Mexico", 52},
        {"Pakistan", 92},
        {"Poland", 48},
        {"South Africa", 27},
        {"Spain", 34},
        {"Thailand", 66},
        {"Ukraine", 380}};

        //print prompt to enter code to search
        printf("Enter an international dialing code: ");
                                                                        1,1           Top
```

clundberg3@gsuad.gsu.edu@snowball:~

```
        {"Spain", 34},
        {"Thailand", 66},
        {"Ukraine", 380}};

        //print prompt to enter code to search
        printf("Enter an international dialing code: ");
        //scan input and store in address of number
        scanf("%d", &number);

        //create variables i for indexing and found for searching
        int i;
        int found = 0;
        //for every structure in the array
        for(i = 0; i <20; i++){
                //if a structure of code member is equal to the int value entered from input
                if(country_codes[i].code == number){
                        //then print the country string literal pointed to by the country member
                        //of the aray of structures
                        printf("The corresponding country is: %s\n", country_codes[i].country);
                        //and assign found to 1
                        found = 1;
                }
        }
        //if found does not equal 1
        if(found != 1){
                //then print the error message
                printf("ERROR : THE INTERNATIONAL DIALING CODE DOES NOT EXIST FOR THIS DATABASE.");
        }
//and exit main method
return 0;
}
                                                                                    72,1          Bot
```

Output of code:

clundberg3@gsuad.gsu.edu@snowball:~

```
[clundberg3@gsuad.gsu.edu@snowball ~]$ vi international-dial_code.c
[clundberg3@gsuad.gsu.edu@snowball ~]$ cc international-dial_code.c
[clundberg3@gsuad.gsu.edu@snowball ~]$ ./a.out
Enter an international dialing code: 62
The corresponding country is: Indonesia
[clundberg3@gsuad.gsu.edu@snowball ~]$ ./a.out
Enter an international dialing code: 380
The corresponding country is: Ukraine
[clundberg3@gsuad.gsu.edu@snowball ~]$ ./a.out
Enter an international dialing code: 41
ERROR : THE INTERNATIONAL DIALING CODE DOES NOT EXIST FOR THIS DATABASE.[clundberg3@gsuad.gsu.edu@sno
[clundberg3@gsuad.gsu.edu@snowball ~]$
```