HW Assignment 4

Marshall Slagle
7/19/18

Program Code Q1

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


struct Box //Declaring a Structure with specific members
{
    char Name[20]; //Declare character array to hold name as string
    int Length; //Declare length member
    int Width; //Declare width member
    int Height; //Declare Height member
    int Area; //Declare Area member
    int Perimeter; //Declare Perimeter member
};

void computeBox(struct Box *Rec); //Method Declaration
void printBox(struct Box Rec); //Method Declaration
int compareBox(struct Box R1, struct Box R2); //Method Declaration


void computeBox(struct Box *Rec) //method to compute area and perimeter of method.
{
    int area = 2*(Rec->Length * Rec->Width) + 2*(Rec->Length * Rec->Height) + 2*(Rec-
>Width * Rec->Height); //Formula calculation to find area of the box given the members
of the structure passed as a parameter by reference through pointer
    int perimeter = 4*(Rec->Length) + 4*(Rec->Width) + 4*(Rec->Height); //Formula
calculation to find perimeter of the box given the member of the structure passed as a
parameter by reference through pointer

    Rec->Area = area; //initializing area member in structure passed reference to
calculated area variable
    Rec->Perimeter = perimeter; //initializing perimeter member in structure passed
reference to calculated area variable
}

void printBox(struct Box Rec) //method to print members of stucture after calculations
{
    printf("\n*****************************\n"); //prints border
    printf("\n%s: \n", Rec.Name); //prints Name of box passed by copy
    printf("\nLength: %d", Rec.Length); //prints Length of box passed by copy
    printf("\nWidth: %d", Rec.Width); //prints Width of box passed by copy
    printf("\nHeight: %d", Rec.Height); //prints Height of box passed by copy
    printf("\nArea: %d", Rec.Area); //prints Area of box passed by copy
    printf("\nPerimeter: %d", Rec.Perimeter); //prints Perimeter of box passed by copy
    printf("\n*****************************\n"); //prints border
```

```
}

int compareBox(struct Box R1, struct Box R2) //method to return comparison value based
on the area of the two boxes
{
    if(R1.Area > R2.Area) //returns 1 if box 1 is larger then box 2
    {
        return 1; //returns 1 to main method
    }
    if(R1.Area == R2.Area) //returns 0 if both boxes are equal
    {
        return 0; //returns 0 to main method
    }
    else //else statement to return value if box 1 is less then box 2
    {
        return -1; //returns -1 to main method
    }
}


int main() //main method
{

    struct Box R1 = {"Rec1", 10, 3, 4, 0, 0}; //initializes R1 of type structure type
box

    struct Box R2 = {"Rec2", 15, 2, 5, 0, 0}; //initializes R2 of type structure type
box

    computeBox(&R1); //calling computebox method to run caclulations, while sending R1
by reference with address
    computeBox(&R2); //calling computebox method to run caclulations, while sending R2
by reference with address

    int compare = compareBox(R1, R2); //calling comparebox method to compare both
boxes and stores return value to compare variable

    printBox(R1); //calls printbox method to print members of structure for R1

    printf("\nComparison value between the boxes: %d\n", compare); //prints return
value of comparison

    printBox(R2); //calls printbox method to print members of stucture for R2
```

```
    //used to demostrate different test
    struct Box R3 = {"Rec3", 5, 10, 2, 0, 0}; //initializes R3 of type structure type
box

    struct Box R4 = {"Rec4", 10, 2, 5, 0, 0}; //initializes R4 of type structure type
box

    computeBox(&R3); //calling computebox method to run caclulations, while sending R3
by reference with address
    computeBox(&R4); //calling computebox method to run caclulations, while sending R4
by reference with address

    compare = compareBox(R3, R4); //calling comparebox method to compare both boxes
and stores return value to compare variable

    printBox(R3); //calls printbox method to print members of structure for R3

    printf("\nComparison value between the boxes: %d\n", compare); //prints return
value of comparison

    printBox(R4); //calls printbox method to print members of stucture for R4


    //used to demostrate different test
    struct Box R5 = {"Rec5", 133, 22, 55, 0, 0}; //initializes R5 of type structure
type box

    struct Box R6 = {"Rec6", 10, 3, 18, 0, 0}; //initializes R6 of type structure type
box
    computeBox(&R5); //calling computebox method to run caclulations, while sending R3
by reference with address
    computeBox(&R6); //calling computebox method to run caclulations, while sending R4
by reference with address

    compare = compareBox(R5, R6); //calling comparebox method to compare both boxes
and stores return value to compare variable

    printBox(R5); //calls printbox method to print members of structure for R5

    printf("\nComparison value between the boxes: %d\n", compare); //prints return
value of comparison

    printBox(R6); //calls printbox method to print members of stucture for R6

}
```

Program Screenshot Output Q1

```
****************************

Rec1:

Length: 10
Width: 3
Height: 4
Area: 164
Perimeter: 68
****************************

Comparison value between the boxes: -1

****************************

Rec2:

Length: 15
Width: 2
Height: 5
Area: 230
Perimeter: 88
****************************

****************************

Rec3:

Length: 5
Width: 10
Height: 2
Area: 160
Perimeter: 68
****************************

Comparison value between the boxes: 0

****************************

Rec4:

Length: 10
Width: 2
Height: 5
Area: 160
Perimeter: 68
****************************

****************************

Rec5:

Length: 133
Width: 22
Height: 55
Area: 22902
Perimeter: 840
****************************

Comparison value between the boxes: 1

****************************

Rec6:

Length: 10
Width: 3
Height: 18
Area: 528
Perimeter: 124
****************************
```

- Initial run.
- The screenshot above shows the full output on the initial run.
- The programs runs through the provided data and extra data added to demonstrate different tests.

```
*****************************

Rec1:

Length: 10
Width: 3
Height: 4
Area: 164
Perimeter: 68
*****************************

Comparison value between the boxes: -1

*****************************

Rec2:

Length: 15
Width: 2
Height: 5
Area: 230
Perimeter: 88
*****************************
```

- Shows top portion of code.
- This is the output with the provided data.
- The output prints each member of the structure in order.
- The program separates the data with star borders.
- The program prints the value from the comparison method in-between the data from the two boxes.

```
****************************

Rec3:

Length: 5
Width: 10
Height: 2
Area: 160
Perimeter: 68
****************************

Comparison value between the boxes: 0

****************************

Rec4:

Length: 10
Width: 2
Height: 5
Area: 160
Perimeter: 68
****************************
```

- Show the middle portion of the output
- The output prints each member of the structure in order.
- The program separates the data with star borders.
- The program prints the value from the comparison method in-between the data from the two boxes.

```
*****************************

Rec5:

Length: 133
Width: 22
Height: 55
Area: 22902
Perimeter: 840
*****************************

Comparison value between the boxes: 1

*****************************

Rec6:

Length: 10
Width: 3
Height: 18
Area: 528
Perimeter: 124
*****************************
```

- Show the bottom portion of the output
- The output prints each member of the structure in order.
- The program separates the data with star borders.
- The program prints the value from the comparison method in-between the data from the two boxes.

Program Output Q1

```
*****************************

Rec1:

Length: 10
Width: 3
Height: 4
Area: 164
Perimeter: 68
*****************************

Comparison value between the boxes: -1

*****************************

Rec2:

Length: 15
Width: 2
Height: 5
Area: 230
Perimeter: 88
*****************************

*****************************

Rec3:

Length: 5
Width: 10
Height: 2
Area: 160
Perimeter: 68
*****************************

Comparison value between the boxes: 0

*****************************

Rec4:
```

Length: 10
Width: 2
Height: 5
Area: 160
Perimeter: 68
****

****

Rec5:

Length: 133
Width: 22
Height: 55
Area: 22902
Perimeter: 840
****

Comparison value between the boxes: 1

****

Rec6:

Length: 10
Width: 3
Height: 18
Area: 528
Perimeter: 124
****

Program Code Q2

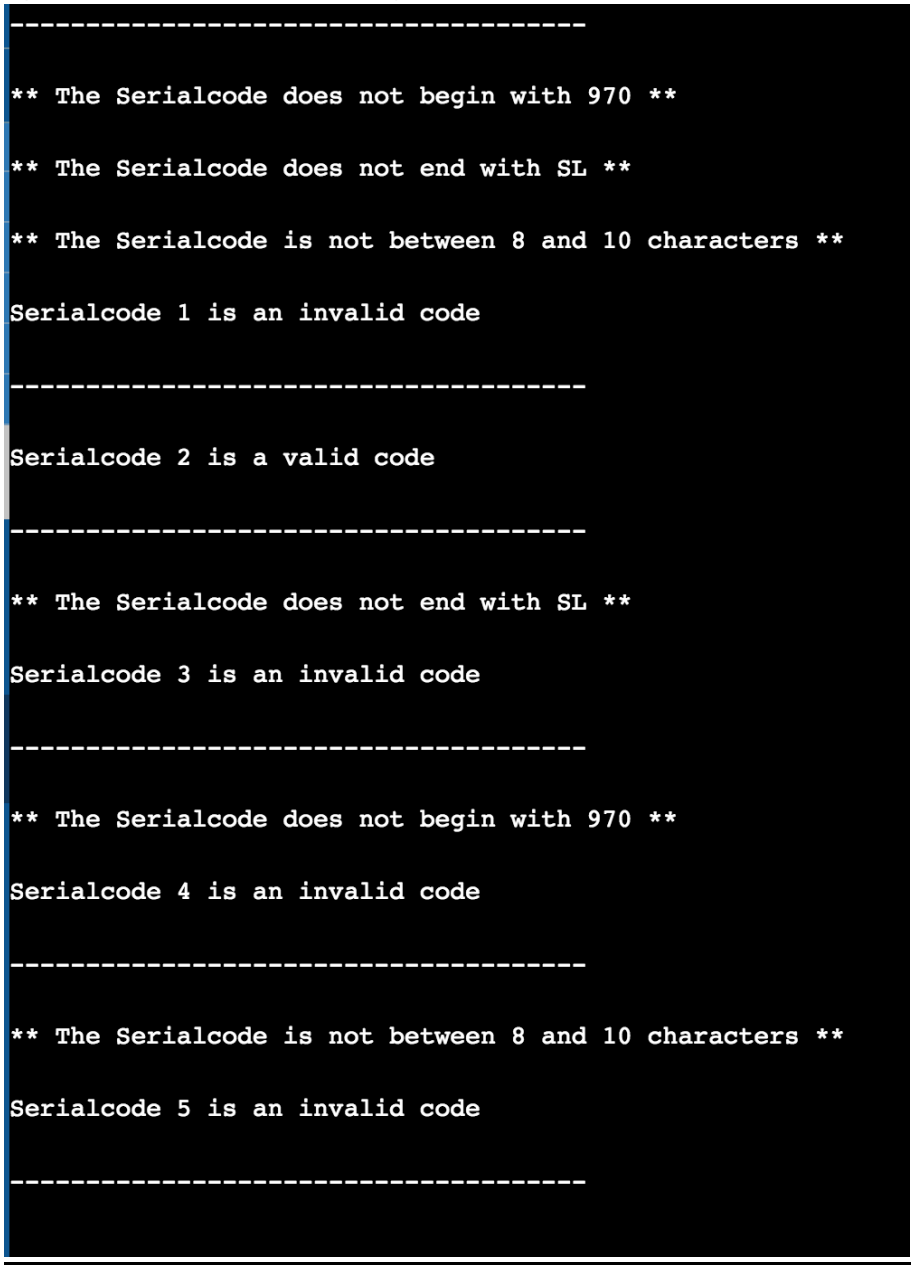
```c
#include <stdio.h>
#include <string.h>
#include <stdbool.h>

bool isValid(char serialcode[]); //Method Declaration


bool isValid(char serialcode[]) //method to determine if serialcode is acceptable
{
    int length = strlen(serialcode); //calculates length of seerialcode string and
initializes it to length variable
    char endL1 = serialcode[length-2]; //finds 2nd to last character in array and
initializes to variable
    char endL2 = serialcode[length-1]; //finds last character in array and initializes
to variable
    int comp = strncmp(serialcode, "970", 3); //compares first 3 characters in array
and initializes variable to compare value

    bool lengthB; //declare boolean variable to store truthful factor of length
    bool endB; //declare boolean variable to store truthful factor of last 2
characters
    bool compB; //declare boolean variable to store truthful factor of first 3
characters

    if(length<=10 && length>=8) //initializes lengthB to true if length is between 8
and 10
    {
        lengthB = true; //initializes lengthB to true
    }
    else
    {
        lengthB = false; //initializes lengthB to false if above statement is not true
    }

    if(endL1=='S' && endL2=='L') //initializes endB to true of last 2 characters are
"SL"
    {
        endB = true; //initializes endB to true
    }
    else
    {
        endB = false; //initializes endB to false if above statement is not true
    }
```

```
    if(comp==0) //initializes compB to true if comparison value is equal to 0
    {
        compB = true; //initializes compB to true
    }
    else
    {
        compB = false; //initializes compB to false if above statement is not true
    }

    if(lengthB == true && endB == true && compB == true) //returns true value to main
method if lengthB, endB, and compB are all true
    {
        return true; //returns true to main method
    }
    if(compB == false) //if compB is false will print error message for the first 3
characters
    {
        printf("\n** The Serialcode does not begin with 970 **\n"); //prints error
message
    }
    if(endB == false) //if endB is false will print error message for the last 2
characters
    {
        printf("\n** The Serialcode does not end with SL **\n"); //prints error
message
    }
    if(lengthB == false) //if lengthB is false will print error message for the length
of Serialcode
    {
        printf("\n** The Serialcode is not between 8 and 10 characters **\n");
//prints error message
    }
    if(lengthB == false || endB == false || compB == false) //if any boolean
determinant variable is false will return false to main method
    {
        return false; //returns false to main method
    }
    else //else statment to provide compiler with option if non of the above
statements are truthful in any fashion
    {
        printf("\n** An error has occured the Serialcode could not be confirmed for
errors **\n"); //prints error message for error in determination
        return false; //returns false do to inability to determine if Serialcode is
correct
    }

}
```

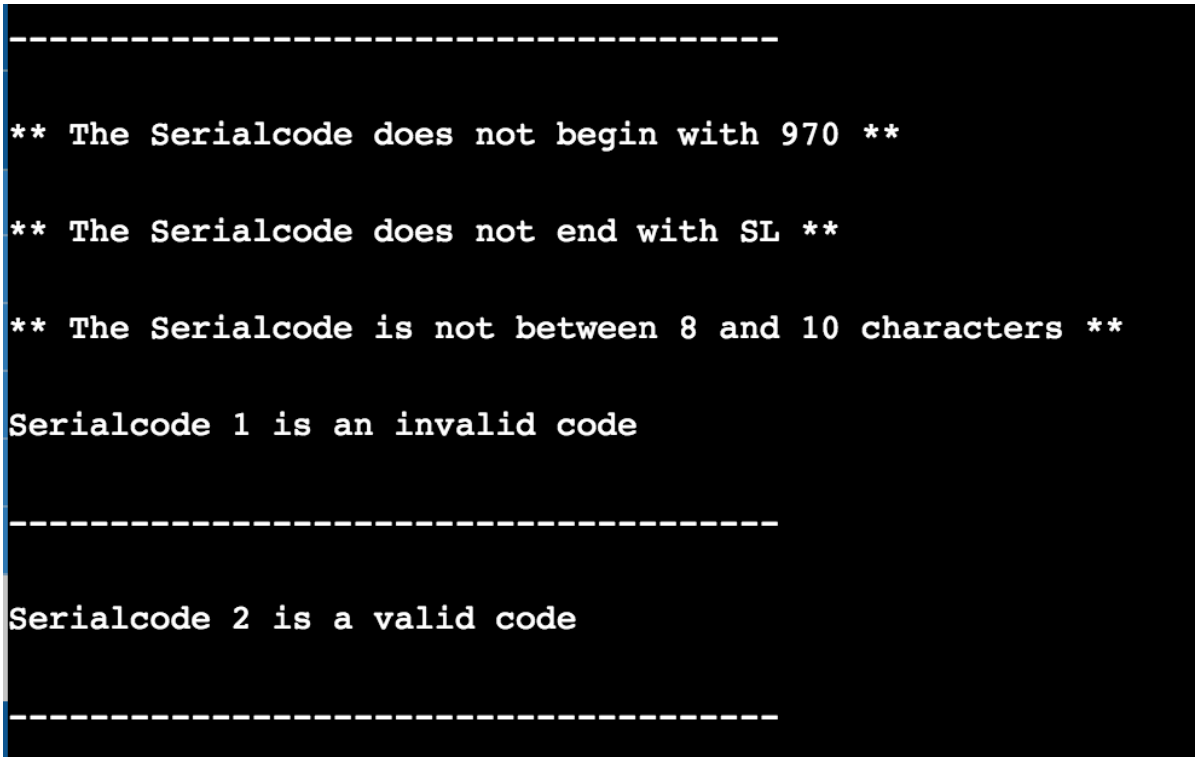
```
int main() //main method
{
    char serialcode1[] = "89238498723498KL"; //initializes string to character array
called serialcode1
    char serialcode2[] = "97083498SL"; //initializes string to character array called
serialcode2

    printf("\n---------------------------------------\n"); //prints border
    if(isValid(serialcode1)) //if statement to determine if return value is truthful
    {
        printf("\nSerialcode 1 is a valid code\n"); //prints successful message. That
serialcode is valid
    }
    else
    {
        printf("\nSerialcode 1 is an invalid code\n"); //prints message that
serialcode is invalid
    }

    printf("\n---------------------------------------\n"); //prints border

    if(isValid(serialcode2)) //if statement to determine if return value is truthful
    {
        printf("\nSerialcode 2 is a valid code\n"); //prints successful message. That
serialcode is valid
    }
    else
    {
        printf("\nSerialcode 2 is an invalid code\n"); //prints message that
serialcode is invalid
    }

    printf("\n---------------------------------------\n"); //prints border



    //used to demonstrate code with different test
    char serialcode3[] = "97083498AL"; //initializes string to character array called
serialcode3 -- used to test code

    if(isValid(serialcode3)) //if statement to determine if return value is truthful
    {
        printf("\nSerialcode 3 is a valid code\n"); //prints successful message. That
serialcode is valid
    }
    else
    {
```

```
        printf("\nSerialcode 3 is an invalid code\n"); //prints message that
serialcode is invalid
    }

    printf("\n---------------------------------------\n"); //prints border



    //used to demonstrate code with different test
    char serialcode4[] = "17083498SL"; //initializes string to character array called
serialcode3 -- used to test code

    if(isValid(serialcode4)) //if statement to determine if return value is truthful
    {
        printf("\nSerialcode 4 is a valid code\n"); //prints successful message. That
serialcode is valid
    }
    else
    {
        printf("\nSerialcode 4 is an invalid code\n"); //prints message that
serialcode is invalid
    }

    printf("\n---------------------------------------\n"); //prints border


    //used to demonstrate code with different test
    char serialcode5[] = "970SL"; //initializes string to character array called
serialcode3 -- used to test code

    if(isValid(serialcode5)) //if statement to determine if return value is truthful
    {
        printf("\nSerialcode 5 is a valid code\n"); //prints successful message. That
serialcode is valid
    }
    else
    {
        printf("\nSerialcode 5 is an invalid code\n"); //prints message that
serialcode is invalid
    }

    printf("\n---------------------------------------\n"); //prints border


}
```

Program Screenshot Output Q2



- Initial run.
- The screenshot above shows the full output on the initial run.
- The programs runs through the provided data and extra data added to demonstrate different tests.



- Shows top portion of code.
- This is the output with the provided data.
- The program separates the data with dash borders.
- The program prints any errors before the validation message.
- The program will print all errors that are found.

```
-------------------------------------

** The Serialcode does not end with SL **

Serialcode 3 is an invalid code

-------------------------------------

** The Serialcode does not begin with 970 **

Serialcode 4 is an invalid code

-------------------------------------

** The Serialcode is not between 8 and 10 characters **

Serialcode 5 is an invalid code

-------------------------------------
```

- Show the bottom portion of the output
- The program separates the data with dash borders.
- The program prints any errors before the validation message.
- The program will print all errors that are found.
- Each serial code has only a single error to demonstrate the programs ability to determine each individual error.

Program Output Q2

** The Serialcode does not begin with 970 **

** The Serialcode does not end with SL **

** The Serialcode is not between 8 and 10 characters **

Serialcode 1 is an invalid code

Serialcode 2 is a valid code

** The Serialcode does not end with SL **

Serialcode 3 is an invalid code

** The Serialcode does not begin with 970 **

Serialcode 4 is an invalid code

** The Serialcode is not between 8 and 10 characters **

Serialcode 5 is an invalid code
