

Front Page

Mîț Alexia Teodora

Bachelor's Computer Science Program

2024-2025 Academic Year

NHL Stenden University of Applied Sciences

Personal email: alexia2207@yahoo.com

School email: alexia.mit@student.nhlstenden.com

Number of EC's: 0/180

The background is a dark blue gradient. In the top left, there are three interlocking gears: a yellow one, a light blue one, and a larger orange one with a textured surface. Blue circuit lines with dots at the ends connect these gears and extend across the top. In the center, a laptop is shown from a slightly elevated front perspective. Its screen is a bright cyan color and displays the word "Portfolio" in a large, white, bold, sans-serif font. The laptop's keyboard is dark grey with light grey keys. In the bottom right corner, there is a graphic of a gold-colored microchip. To the right of the chip, there are several lines of binary code (0s and 1s) in white and light blue. At the bottom right, there are several curved, parallel lines in a light green color, resembling a circuit board trace.

Portfolio

010
010
101

0101001
11111101
0101010
010010010101001
11101011111101
0100100101001

Table of Contents

Front Page	1
Table of Contents	3
Introduction	5
Week 0	6
Scratch	6
Cutebot	10
Week 0 - Self reflection	12
Hello, It's me assignment	14
Mario Less Comfortable	15
Mario More Comfortable.....	17
Week 1 – Mario – Embedded Systems	22
Mario Less Comfortable code	22
Mario More Comfortable code	24
Week 1 - Self reflection	26
Week 2 – Arrays	27
Scrabble.....	27
Readability	30
Substitution.....	33
Week 2 – Morse Code	37
Week 2 - Self reflection	42
Week 3 – Algorithms	43
Sort.....	43
Plurality.....	45
Runoff	49
Week 3 - Traffic light project with pedestrian crossing.....	56
Week 3 - Self reflection	61
Week 4 – Memory	62
Volume	62
Filter-less	65
Filter-more	70
Recover.....	76

Week 4 – Displaying Uptime	80
Week 4 – Self reflection.....	84
Week 5 – Data Structures.....	85
Inheritance	85
Week 5 – Jumbo Clock	90
Week 5 – Self reflection.....	96
Week 6 – Speller.....	97
Week 6 – Weather Station	101
Week 6 – Self reflection.....	109
GitHub project – using PlatformIO	110
Set up guides	114
Visual Studio Code.....	114
GitHub for Desktop	117
Git.....	119
Docker in Visual Studio Code	122
PlatformIO IDE.....	123

Introduction

My name is Mîț Alexia Teodora, and I am 19 years old. I moved from Romania to The Netherlands by myself so I can follow the Computer Science program at NHL Stenden University of Applied Sciences. I chose to come here, because it has a different way of teaching than the Universities from my country. In my opinion, the decision that I made will give me more opportunities to pursue my future career.

This portfolio will show in detail my progress and the skills I gained while following the chosen program.

In addition, my GitHub name is Alexia220700.

Week 0

Scratch

- Create a Scratch Game: <https://scratch.mit.edu>
- Create an account
- Your project must use at least two sprites.
- Your project must have at least three scripts total (i.e., not necessarily three per sprite).
- Your project must use at least one conditional, at least one loop, and at least one variable.
- Your project must use at least one custom block that you have made yourself (via **Make a Block**), which must take at least one input.
- Your project should be more complex than most of those demonstrated in the lecture.

Logbook from date to date

Day	Logbook of work
Tuesday, 03.09.2024	I started to work on my project. I made a story at the beginning of the game, but I gave up on it in the end. In addition, I worked on the first two Sprites, the main ones, the shark and the crab.
Wednesday	I started to work on the other Sprites. For me, the fishes were easier to make, I chose the way they move, the messages and which ones to multiply.
Thursday	On Thursday I tried to perfect the game. I decided to add a song to it and a timer.
Friday	I handed in my Scratch Game.

I made a game intitulated “Megalodon game”:

<https://scratch.mit.edu/projects/1063438227>

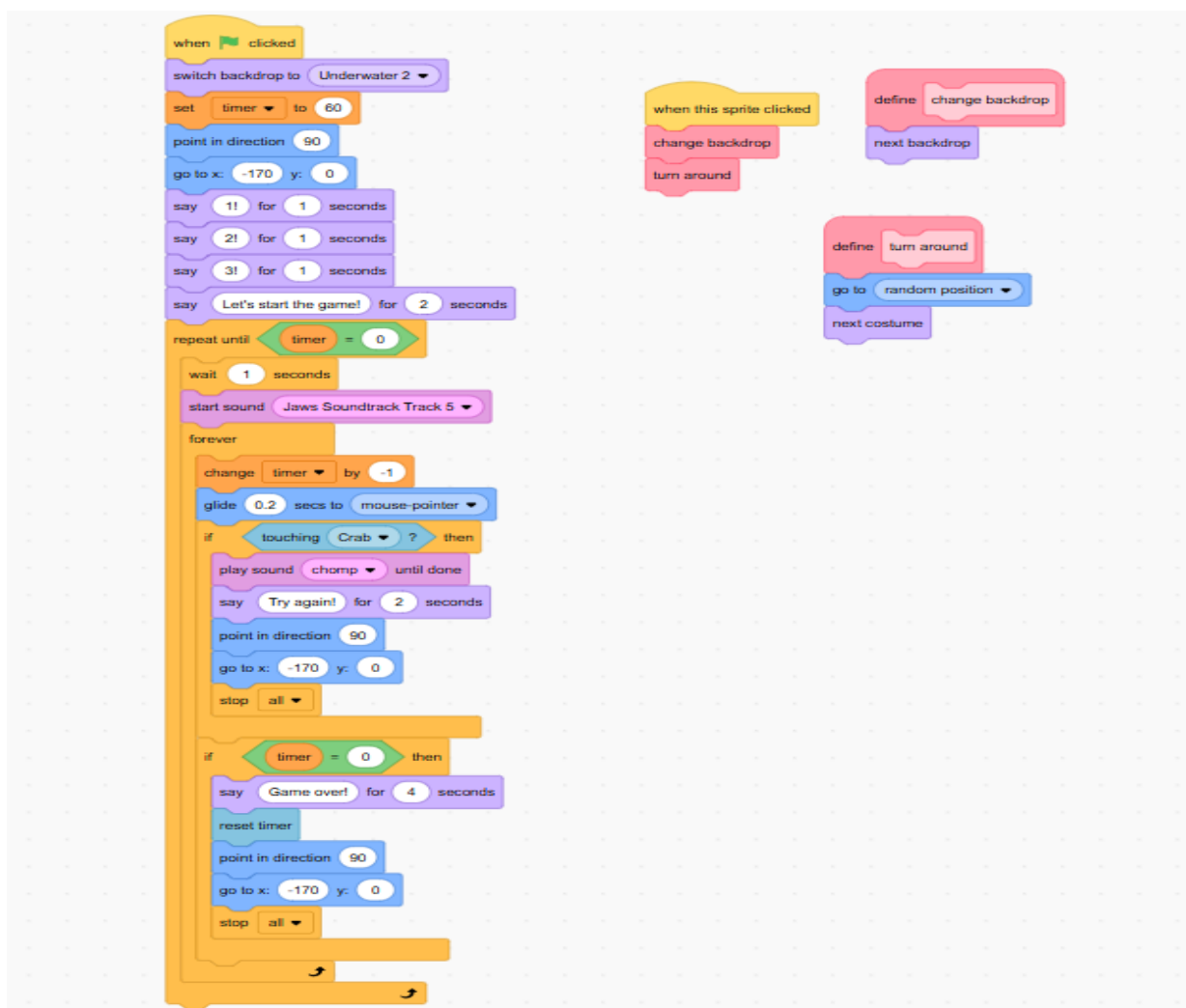
How do you play it?

At the beginning there is a countdown, then the game starts. I chose the Jaws movie theme song as it is an underwater game.

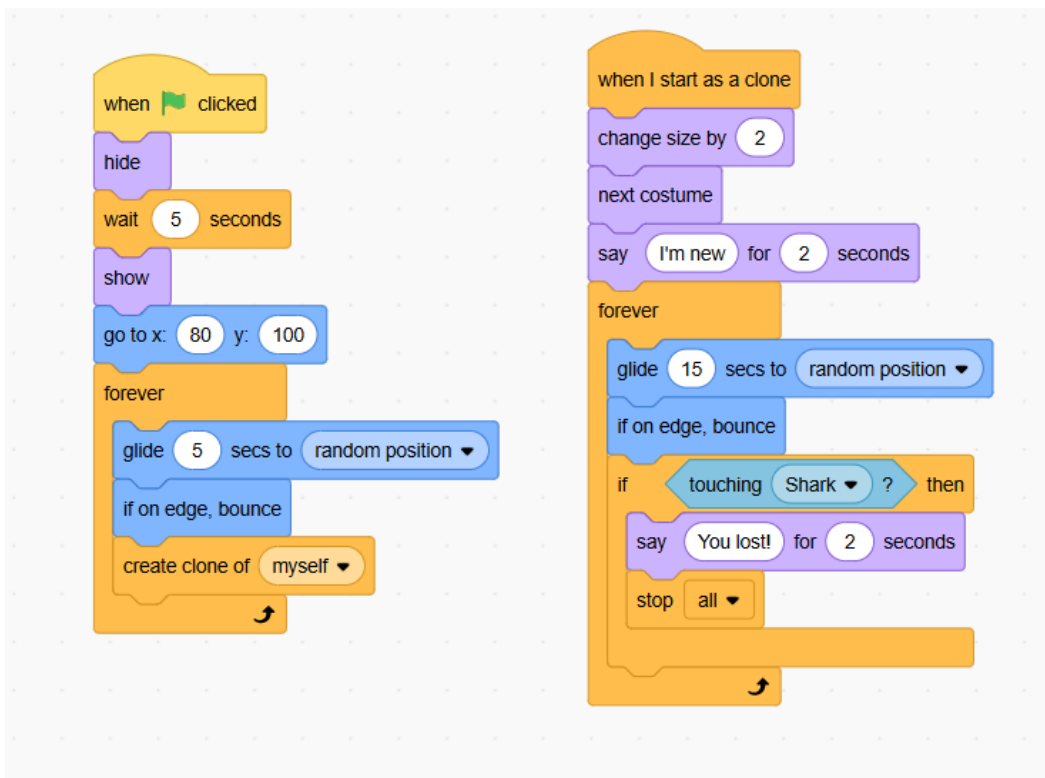
As a player, you must catch the fish while the timer is running down. If the Shark Sprite touches the fish they'll show a message.

If the Shark Sprite touches one of the Crab Sprites or the timer reaches zero, the game will end and go back to how it was at the beginning.

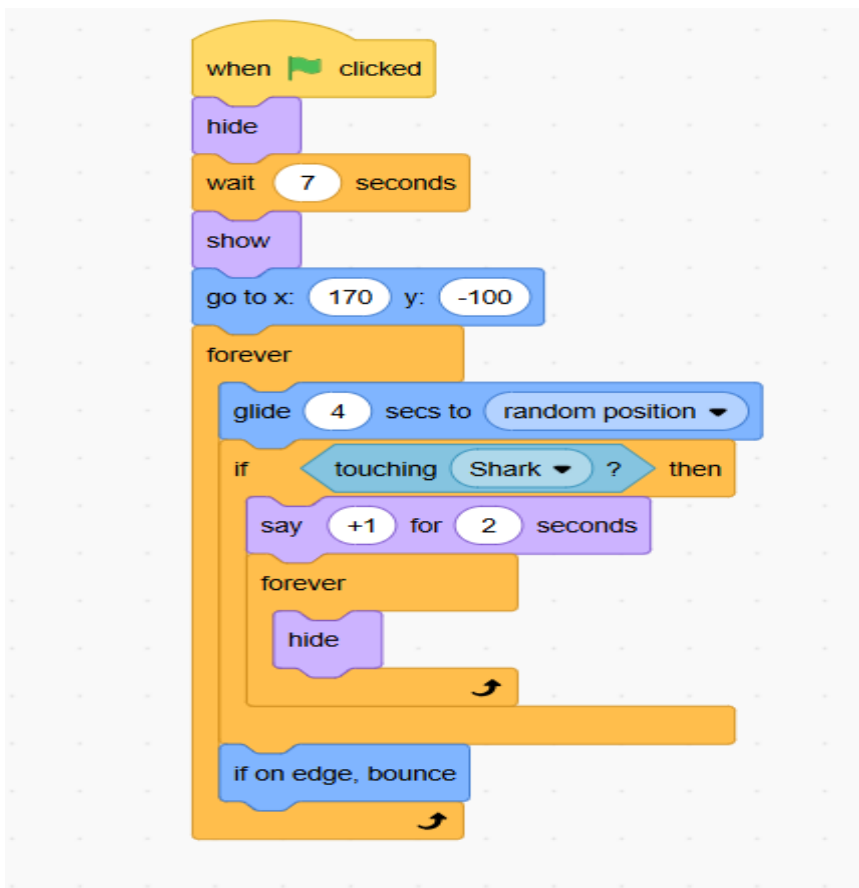
This is the code for one of the Sprites (the Shark):



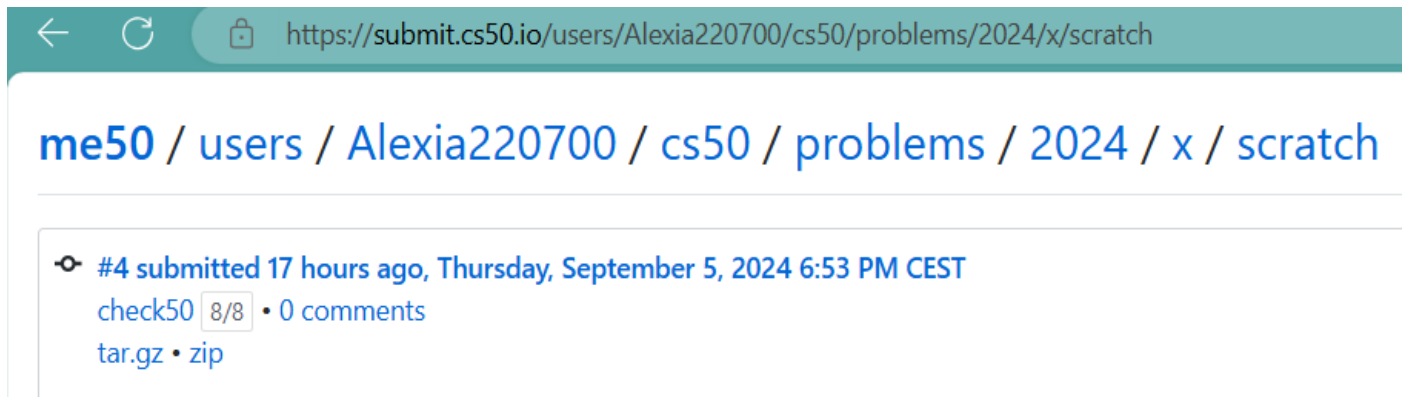
This is the code for another Sprite (the Crab):



This is the code for another Sprite (the Pufferfish):



This screenshot shows the Harvard score with the title of the program and my name:

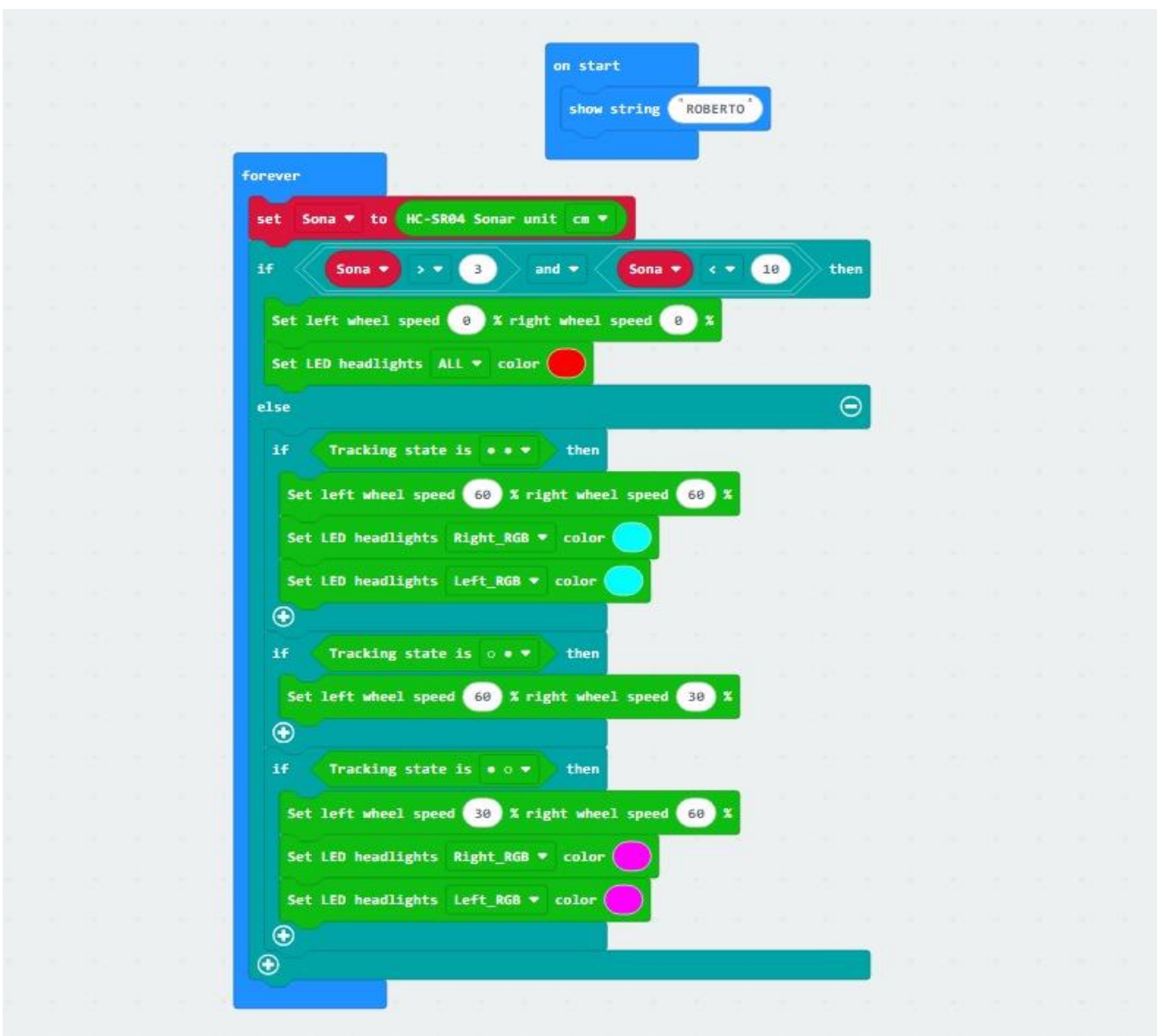


<https://submit.cs50.io/check50/3c920f8941c4868d9afd76304f797a2cd6e1ec32>

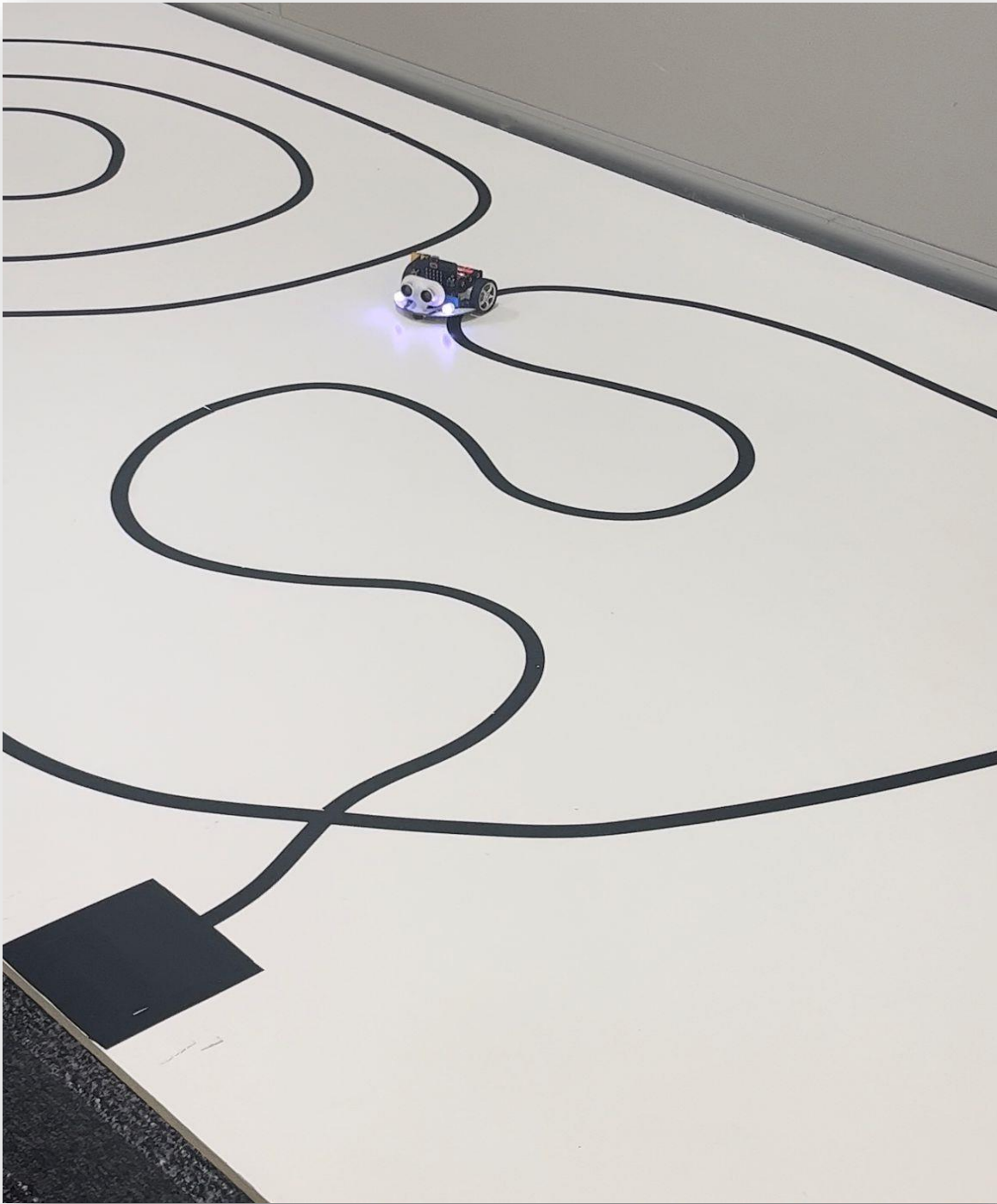
Cutebot

For the Embedded Systems class we had to program a Cutebot that follows a black line while keeping distance from the other Cutebots. In addition, if the Cutebot encounters an obstacle, it has to stop. If the obstacle is removed from the robot's way, the Cutebot has to start to follow the black line again.

This screenshot shows the code for our Cutebot, named Roberto:



This is a picture of our Cutebot on the track:



Week 0 - Self reflection

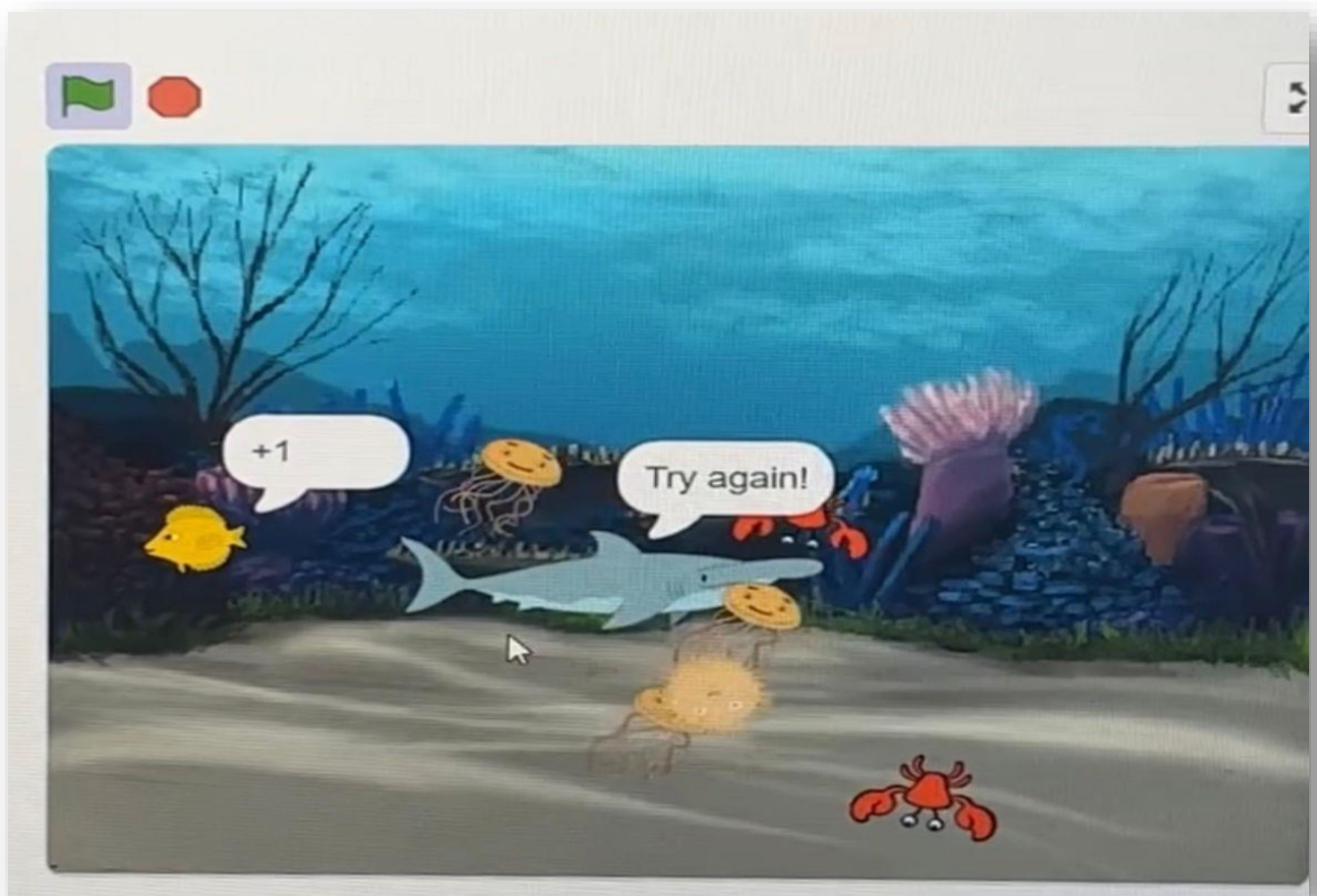
During the first week of University, I tried to get used to student life and how everything works. It was easy for me, thanks to all the nice people I've met, especially my colleagues.

I found the Computer Science class fun. I enjoyed creating the first assignment, because it let me use my creativity and develop my skills in programming. For my assignment, I decided to make an underwater game. The shark had to catch the fish, without touching the crabs. I also added a 60 second timer and music.

The Embedded Systems class was a bit more demanding as we had to program a Cutebot. In my opinion, it was a good choice to work in groups, because we could share our opinions with each other and combine them. The small Cutebots had to follow the black line, keep their distance from the other Cutebots, get back on the black line if they get lost, show a message and stop when they meet an obstacle in their path.

In addition, the Dutch class is important to us, international students. By attending the class, we learn new words, how to communicate and integrate into the community. Everyone was happy to be there and tried their best to retain the information.

This is a picture of my Scratch game:



Week 1 – CS50x: C

Hello, It's me assignment

In a file called `hello.c`, in a folder called `me`, implement a program in C that prompts the user for their name and then says hello to that user. For instance, if the user's name is Adele, your program should print hello, Adele\n!

This is the code for the assignment:

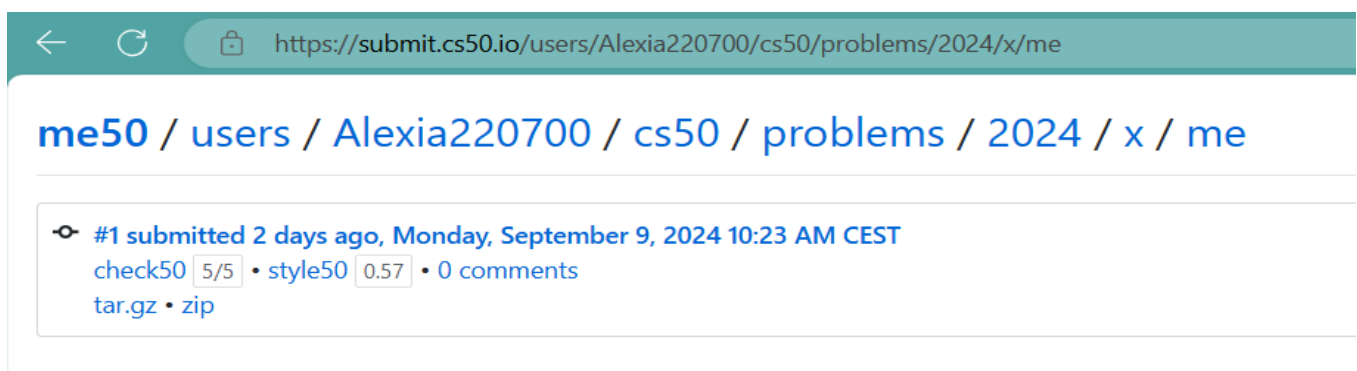
```
#include<stdio.h>
#include<cs50.h>

int main (void)
{
    string name = get_string("What is your name?");
    printf("Hello, %s\n", name);
}
```

Logbook from date to date:

Day	Logbook of work
Monday, 09.09.2024	Done the assignment.

The Harvard score with the title of the program and my name:



The screenshot shows a web browser with the URL <https://submit.cs50.io/users/Alexia220700/cs50/problems/2024/x/me>. The page displays the submission details for the 'me' problem. It shows that the user has submitted the solution 2 days ago, on Monday, September 9, 2024, at 10:23 AM CEST. The submission is marked as '#1 submitted'. The score for the submission is 0.57, with a check50 score of 5/5 and a style50 score of 0.57. There are 0 comments. The submission is available in tar.gz and zip formats.

<https://submit.cs50.io/check50/9f5c53333b98888f961af7b5404ac933a68bff36>

Mario Less Comfortable

In a file called `mario.c` in a folder called `Mario-less`, implement a program in C that recreates a pyramid, using hashes (#) for bricks.

This is the code for the assignment:




```
#include <cs50.h>
#include <stdio.h>
int main(void)
{
    int height;
    do
    {
        height = get_int("Height:"); // prompt user for height
    }
    while (height < 1 || height > 8);

    for (int i = 0; i < height; i++)
    {
        for (int j = 0; j < height; j++)
        {
            if (i + j < height - 1)
                printf(" "); // print spaces
            else
                printf("#"); // print hashes
        }
        printf("\n"); // endl
    }
}
```


Logbook from date to date

Day	Logbook of work
Monday, 09.09.2024	I started working on the project.
Tuesday, 10.09.2024	Finished the project.
Wednesday, 11.09.2024	Submitted the project on submit50

The Harvard score with the title of the program and my name:

   <https://submit.cs50.io/users/Alexia220700/cs50/problems/2024/x/mario/less>

[me50](#) / [users](#) / [Alexia220700](#) / [cs50](#) / [problems](#) / [2024](#) / [x](#) / [mario](#) / [less](#)

 **#1 submitted 5 hours ago, Wednesday, September 11, 2024 10:29 AM CEST**
[check50](#) 10/10 • [style50](#) 1.00 • [0 comments](#)
[tar.gz](#) • [zip](#)

<https://submit.cs50.io/check50/08986c1d89fb6425a0613f89060588bc642f7688>

Mario More Comfortable

In a file called `mario.c` in a folder called `Mario-more`, implement a program in C that recreates a pyramid, using hashes (#) for bricks.

This is the code for the assignment :

```
#include <stdio.h>




int main(void)
{
    int height;
    do
    {
        height = get_int("Height:"); // prompt user for height
    }
    while (height < 1 || height > 8);

    for (int i = 0; i < height; i++) //for height
    {
        for (int j = 0; j < height + i + 3; j++) //for width
        {
            if (i + j < height - 1 || j == height || j == height + 1)
                printf(" "); // print spaces
            else
                printf("#"); // print hashes
        }
        printf("\n");
    }
}
```


Logbook from date to date

Day	Logbook of work
Wednesday, 11.09.2024	Done the assignment and submitted it on submit50.

The Harvard score with the title of the program and my name:

 <https://submit.cs50.io/users/Alexia220700/cs50/problems/2024/x/mario/more>

[me50](#) / [users](#) / [Alexia220700](#) / [cs50](#) / [problems](#) / [2024](#) / [x](#) / [mario](#) / [more](#)

 [#1 submitted 2 hours ago, Wednesday, September 11, 2024 1:40 PM CEST](#)
[check50](#) • [style50](#) • [0 comments](#)
[tar.gz](#) • [zip](#)

<https://submit.cs50.io/check50/6f202f4ab078edae8d864b44e47f19d3c10748fc>

Credit

In the file called `credit.c` in the `credit` directory, write a program that prompts the user for a credit card number and then reports whether it is a valid American Express, MasterCard, or Visa card number, per the definitions of each's format herein. So that we can automate some tests of your code, we ask that your program's last line of output be `AMEX\n` or `MASTERCARD\n` or `VISA\n` or `INVALID\n`, nothing more, nothing less.

This is the code for the assignment :

```
#include <cs50.h>
#include <stdio.h>
int get_digit_sum(long number);
bool is_valid_credit_card(long number);

int main(void)
{
    long number = get_long("Number: ");

    if (is_valid_credit_card(number))
    {
        //length starts from zero
        int length = 0;
        long aux = number;

        while (aux != 0) //get the lenght of card number
        {
            aux = aux / 10;
            length++;
        }

        // checks if the lenght and first two digits *or first digit for VISA* are correct

        if (length == 15 && (number / 100000000000000 == 34 || number / 100000000000000 ==
37))
        {
            printf("AMEX\n");
        }
        else if (length == 16 && (number / 1000000000000000 >= 51 && number /
1000000000000000 <= 55))
        {

```

```

        printf("MASTERCARD\n");
    }
    else if ((length == 13 || length == 16) &&
        (number / 1000000000000 == 4 || number / 1000000000000000 == 4))
    {
        printf("VISA\n");
    }
    else
    {
        printf("INVALID\n");
    }
}
else
{
    printf("INVALID\n");
}
}

int get_digit_sum(long number)
{
    int sum = 0;
    bool alternate = false;

    while (number > 0)
    {
        int digit = number % 10;
        number = number / 10;

        if (alternate)
        {
            digit = digit * 2; // multiplies every other digit by 2, starting with the
number's
                                // second-to-last digit
            if (digit > 9)
            {
                digit = digit - 9;
            }
        }

        sum = sum + digit; // makes a sum of the digits
        alternate = !alternate;
    }
    return sum;
}

```

```
bool is_valid_credit_card(long number) // checks if the credit card number is valid,
sum%10 == 0
{
    int sum = get_digit_sum(number);

    return (sum % 10 == 0);
}
```

Logbook from date to date

Day	Logbook of work
Wednesday, 12.09.2024	I started working on the assignment. I did it in C++, because it was easier for me to understand how it works in that way.
Thursday, 13.09.2024	I tried to translate the code from C++ to C. I also had a problem with the sum of the digits and tried to fix it.
Saturday, 14.09.2024	Done the assignment and submitted it on submit50.

The Harvard score with the title of the program and my name:

[me50](#) / [users](#) / [Alexia220700](#) / [cs50](#) / [problems](#) / [2024](#) / [x](#) / [credit](#)

🔗 #1 submitted a day ago, Saturday, September 14, 2024 7:35 PM CEST
[check50](#) 17/17 • [style50](#) 1.00 • 0 comments
[tar.gz](#) • [zip](#)

<https://submit.cs50.io/check50/47c36e1f3d887c8d47c5b4292b9968c1b2c273f3>

Week 1 – Mario – Embedded Systems

Mario Less Comfortable code

In a file called mario.c in a folder called Mario-less, implement a program in Arduino that recreates a pyramid, using hashes (#) for bricks.

This is the code for the assignment:

```
int height = 0;

void setup()
{
    //code runs once
    Serial.begin(9600);
    delay(500);

    Serial.println("Choose height between 1-8: "); //asks for height

    while(true) //checks to see if height is less than 1 or more than 8
    {
        if(Serial.available() > 0) //checks if the number is more than 0, positive
        {
            height = Serial.parseInt(); // read integer from serial input and height takes that
value

            Serial.read(); //used to get rid of residual data
                        //and stop the "Choose a number between 1-8: " from showing on the screen
                        //more than once

            if(height >=1 && height <=8) //if the value corresponds to that criteria
            {
                break; //program stops if it finds a correct height
            }
            else
            {
                Serial.println("Choose a number between 1-8: ");
//the program will show this message until the height is correctly chosen
            }
        }
    }
}
```

```

void loop()
{
    //code runs until the end, until the pyramid is done

    for (int i = 0; i < height; i++)
    {
        for (int j = 0; j < height; j++)
        {
            if (i + j < height - 1)
                Serial.print(" "); // print spaces
            else
                Serial.print("#"); // print hashes
        }
        Serial.println();
    }

    while(true) //prevents the code from creating multiple pyramids
    {

    }
}

```

Logbook from date to date

Day	Logbook of work
Saturday, 14.09.2024	Done the assignment.

Mario More Comfortable code

In a file called mario.c in a folder called Mario-more, implement a program in Arduino that recreates a pyramid, using hashes (#) for bricks.

This is the code for the assignment :

```
int height = 0;

void setup()
{
    //code runs once
    Serial.begin(9600);
    delay(500);

    Serial.println("Choose height between 1-8: "); //asks for height

    while(true) //checks to see if height is less than 1 or more than 8
    {
        if(Serial.available() > 0) //checks if the number is more than 0, positive
        {
            height = Serial.parseInt(); // read integer from serial input and height takes
that value

            Serial.read(); //used to get rid of residual data
//and stop the "Choose a number between 1-8: " from showing on the
screen

//more than once

            if(height >=1 && height <= 8) //if the value corresponds to that criteria
            {
                break; //program stops if it finds a correct height
            }
            else
            {
                Serial.println("Choose a number between 1-8: ");
//the program will show this message until the height is correctly chosen
            }
        }
    }
}

void loop()
```



```

{
    //code runs until the end, until the pyramid is done

for (int i = 0; i < height; i++)
{
    for (int j = 0; j < height + i + 3; j++)
    {
        if (i + j < height - 1 || j == height || j == height + 1)
            Serial.print(" "); // print spaces
        else
            Serial.print("#"); // print hashes
    }
    Serial.println();
}
while(true) //prevents the code from creating multiple pyramids
{

}
}

```

Logbook from date to date

Day	Logbook of work
Saturday, 14.09.2024	Done the assignment.

Week 1 - Self reflection

Week 1 was interesting and easy, because I enjoyed the assignments given.

During one of our Computer Science classes we did Hello, It's me assignment. This was important for us to get used to the basics of C programming language.

I made Mario Less Comfortable and Mario More Comfortable. I chose to do both so I could understand how to create the first pyramid, then implement it in the Mario More, with two pyramids. The Mario Less took me three days to make, then I did Mario More in one day.

The Credit assignment was a challenge for me, so I made it in C++ first. Thanks to that, I could understand better how it should work in C programming language.

For the Embedded Systems class, we had to implement Mario More in Arduino programming language. Again, I did both assignments, Mario Less and Mario More. I found it easy, and I did the assignments the same day.

During my free time, I also worked on my portfolio. I tried to keep it updated with my assignments.

Week 2 – Arrays

Scrabble

In a file called `scrabble.c` in a folder called `scrabble`, implement a program in C that determines the winner of a short Scrabble-like game. Your program should prompt for input twice: once for “Player 1” to input their word and once for “Player 2” to input their word. Then, depending on which player scores the most points, your program should either print “Player 1 wins!”, “Player 2 wins!”, or “Tie!” (in the event the two players score equal points).

This is the code for the assignment:

```
#include <cs50.h>
#include <ctype.h>
#include <stdio.h>
#include <string.h>

// introduces the points for each letter of the alphabet
int POINTS[] = {1, 3, 3, 2, 1, 4, 2, 4, 1, 8, 5, 1, 3, 1, 1, 3, 10, 1, 1, 1, 1, 4, 4, 8, 4, 10};
int compute_sc(string word);

int main(void)
{
    // prompts users for two words, two strings
    string word1 = get_string("Player 1: ");
    string word2 = get_string("Player 2: ");

    // calculates the scores
    int sc1 = compute_sc(word1);
    int sc2 = compute_sc(word2);

    // compares the scores and shows which of the two players wins or if their scores are equal
    if (sc1 > sc2)
    {
        printf("Player 1 wins!\n");
    }
    else if (sc2 > sc1)
```

```

{
    printf("Player 2 wins!\n");
}
else
{
    printf("Tie!\n");
}
}

// every string is an array
// computes the score for each word
int compute_sc(string word)
{
    // score starts from 0
    int sc = 0;

    // strlen calculates the length of the string
    for (int i = 0; i < strlen(word); i++)
    {
        if (word[i] >= 65 && word[i] <= 90)
        {
            // score equal to score plus index in ASCII minus 65
            // 'A' in ASCII is 65, but in POINTS it's 0, only if it's uppercase
            sc = sc + POINTS[word[i] - 'A'];
        }
        else if (word[i] >= 97 && word[i] <= 122)
        {
            // for lowercase, it's minus 97
            sc = sc + POINTS[word[i] - 'a'];
        }
    }
    return sc;
}

```

Logbook from date to date

Day	Logbook of work
Tuesday, 17.09.2024	Done the assignment.

The Harvard score with the title of the program and my name:

[me50](#) / [users](#) / [Alexia220700](#) / [cs50](#) / [problems](#) / [2024](#) / [x](#) / [scrabble](#)

🔑 #1 submitted 7 minutes ago, Tuesday, September 17, 2024 9:20 PM CEST
check50 11/11 • style50 1.00 • 0 comments
[tar.gz](#) • [zip](#)

<https://submit.cs50.io/check50/85858273f10948411bd47f24c74d7daf4d557c02>

Readability

In a file called `readability.c` in a folder called `readability`, you'll implement a program that calculates the approximate grade level needed to comprehend some text. Your program should print as output "Grade X" where "X" is the grade level computed, rounded to the nearest integer. If the grade level is 16 or higher (equivalent to or greater than a senior undergraduate reading level), your program should output "Grade 16+" instead of giving the exact index number. If the grade level is less than 1, your program should output "Before Grade 1".

This is the code for the assignment:

```
#include <cs50.h>
#include <ctype.h>
#include <math.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    // prompts user for text
    string text = get_string("Enter a text: ");

    int letters = 0;
    int words = 1;
    int sentences = 0;

    // using a loop to count words, letters and sentences
    for (int i = 0; i < strlen(text); i++)
    {
        // counting the letters using ASCII table
        if (isalpha(text[i]))
        {
            letters++;
        }
        // counting the sentences
        else if (text[i] == '.' || text[i] == '!' || text[i] == '?')
        {
            sentences++;
        }
        // counting the words
    }
```

```

        else if (isspace(text[i]))
        {
            words++;
        }
    }

    // calculate L and S
    float L = (float) letters / words * 100;
    float S = (float) sentences / words * 100;

    // round the index for the grades
    int index = round(0.0588 * L - 0.296 * S - 15.8);

    // print the grade level
    if (index < 1)
    {
        printf("Before Grade 1\n");
    }
    else if (index >= 16)
    {
        printf("Grade 16+\n");
    }
    else
    {
        printf("Grade %i\n", index);
    }
}

```

Logbook from date to date

Day	Logbook of work
Tuesday, 17.09.2024	Started the assignment. On this day I did most work, but I had a problem with calculating the L and S.
Wednesday, 18.09.2024	I fixed the problem with L and S. Done the assignment.

The Harvard score with the title of the program and my name:

[me50](#) / [users](#) / [Alexia220700](#) / [cs50](#) / [problems](#) / [2024](#) / [x](#) / [readability](#)

🔑 [#1 submitted a few seconds ago, Wednesday, September 18, 2024 9:04 PM CEST](#)
[check50](#) 11/11 • [style50](#) 1.00 • [0 comments](#)
[tar.gz](#) • [zip](#)

<https://submit.cs50.io/check50/4e7c9a4dadf40525867aa98f10f99550fe632edb>

Substitution

In a substitution cipher, we “encrypt” (i.e., conceal in a reversible way) a message by replacing every letter with another letter. To do so, we use a *key*: in this case, a mapping of each of the letters of the alphabet to the letter it should correspond to when we encrypt it. To “decrypt” the message, the receiver of the message would need to know the key, so that they can reverse the process: translating the encrypt text (generally called *ciphertext*) back into the original message (generally called *plaintext*).

This is the code for the assignment:

```
#include <cs50.h>
#include <ctype.h>
#include <stdio.h>
#include <string.h>

int main(int argc, string argv[])
{
    // error message if there are less than 2 or more than 2 arguments
    if (argc != 2)
    {
        printf("./substitution KEY\n");
        return 1;
    }

    // if there are two arguments
    if (argc == 2)
    {
        // get the second argument
        string key = argv[1];

        for (int i = 0; i < 26; i++)
        {
            // checks if all of the inputs are letters
            if (!isalpha(key[i]))
            {
                // outputs an error message if one of the inputs is not a letter
                printf("Key is incorrect. It should only contain letters!\n");
                return 1;
            }
        }
    }
}
```

```

// checks if the characters are different
// first string that starts from the first letter
for (int a = 0; a < strlen(key); a++)
{
    // second string that starts from the second letter
    /*if it started from the first letter too, there would be an error and a[i]
== b[i]*
    for (int b = a + 1; b < strlen(key); b++)
    {
        if (key[a] == key[b])
        {
            // if characters are not different, the program show this error
message
            printf("Key must contain different characters!\n");
            return 1;
        }
    }
}

// checks if the input is the right length
if (strlen(key) != 26)
{
    // if it's not the right length, it will show the error message
    printf("Key must contain 26 characters!\n");
    return 1;
}

// asks for plaintext
string plaintext = get_string("plaintext: \n");

// the program outputs ciphertext: without a newline
printf("ciphertext: ");

// checks if char is lowercase/uppercase/not a letter
for (int c = 0; c < strlen(plaintext); c++)
{
    // checks if it's a letter
    if (isalpha(plaintext[c]))
    {
        // checks if it's lowercase
        if (islower(plaintext[c]))
        {
            // outputs a character
            printf("%c", tolower(key[plaintext[c] - 'a']));
        }
        // checks if it's uppercase
        else if (isupper(plaintext[c]))
        {

```

```

        // outputs a character
        printf("%c", toupper(key[plaintext[c] - 'A']));
    }
}
// if it's not a letter
else
{
    // outputs the rest of the characters
    printf("%c", plaintext[c]);
}
}
}
printf("\n");
return 0;
}

```

Logbook from date to date

Day	Logbook of work
Wednesday, 18.09.2024	I started working on the command-line.
Thursday, 19.09.2024	I worked on the cases when the key is invalid.
Wednesday, 18.09.2024	I checked if the characters are uppercase, lowercase. In addition, I checked if they are letters or other characters.

The Harvard score with the title of the program and my name:

[me50](#) / [users](#) / [Alexia220700](#) / [cs50](#) / [problems](#) / [2024](#) / [x](#) / [substitution](#)

[My Submissions](#)

[My Courses](#)

[Docs](#)

[Log Out](#)

🔑 #1 submitted a few seconds ago, Friday, September 20, 2024 5:40 PM CEST

[check50](#) 18/18 • [style50](#) 1.00 • 0 comments

[tar.gz](#) • [zip](#)

<https://submit.cs50.io/check50/9e086d6894660bfb6d3d468dde03813ccf5a577b>

Week 2 – Morse Code

Create a program that converts letters into Morse code.

This is the code for the assignment:

```
//initializing arrays
char alphabet[26] = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'};

//const char*, because Morse code of each letter is a string
const char* morseCode[26] = {".-", "-...", "-.-.", "-..", ".", "-.-.", "--.", "....",
"..", ".---", "-.-", "-...", "--", "-.", "---", "-.-.", "--.-", "-.", "...", "-", "-.-",
"...-", ".--", "-.-", "-.-", "-.-."};

#define ON HIGH    //defines when LED is on
#define OFF LOW    //defines when LED is off

int LED = 13;      //LED at port 13 / pins for LED
int buzzer = 12;   //pins for buzzer

int dotLength = 100; //length of a dot
int dashLength = dotLength * 3; //length of a dash

//frequencies for tone, buzzer
int dotFrequency = 1000; //1kHz
int dashFrequency = 500; //500Hz

//match the Morse code to the letter
const char* getMorseCode(char letter)
{
    //convert the letter to uppercase if needed
    letter = toupper(letter);

    //find the index of the letter
    for(int i = 0; i < 26; i++)
    {
        if(alphabet[i] == letter)
        {
            //return Morse code corresponding to the letter
            return morseCode[i];
        }
    }
    //return empty string if it doesn't find a match
```

```

    return "";
}

void setup()
{
    // put your setup code here, to run once:
    Serial.begin(9600);
    delay(500);

    //configure LED as output
    pinMode(LED, OUTPUT);

    //configure buzzer as output
    pinMode(buzzer, OUTPUT);

    //read the string input through Serial Monitor
    Serial.println("Enter your message here: ");
}

//configuration for dot
void blinkDot()
{
    // put your main code here, to run repeatedly:
    digitalWrite(LED, ON);

    //reproduce tone for dot
    tone(buzzer, dotFrequency);

    delay (dotLength);
    digitalWrite(LED, OFF);

    //stop tone
    noTone(buzzer);

    delay(dotLength); //space between parts of the same letter
}

//configuartion for dash
void blinkDash()
{
    // put your main code here, to run repeatedly:
    digitalWrite(LED, ON);

    //reproduce tone for dash
    tone(buzzer, dashFrequency);
    delay (dashLength);
}

```

```

digitalWrite(LED, OFF);

//stop tone
noTone(buzzer);

delay(dotLength); //using dotLength so they are the same *delay for dash and dot*
//space between parts of the same letter
}

//blink Morse Code for a letter
void blinkMorseCode(char letter)
{

    //get Morse code for the letter
    const char* code = getMorseCode(letter);

    //print letter
    Serial.print(letter);
    Serial.println(code);

    //goes through the while loop to use the buzzer and LED
    while(*code)
    {
        //if character is a ".", outputs the configuration for dot
        if(*code == '.')
        {
            blinkDot();
        }
        //if character is a "-", outputs the configuration for dash
        else if(*code == '-')
        {
            blinkDash();
        }
        //code goes to the next character
        code++;
    }
    //space between letters is equal to three dots / a dash
    delay(dotLength * 3);
}

void loop()
{
    //check if data is available from Serial Monitor
    if (Serial.available())
    {
        //read the string input
        String input = Serial.readString();
    }
}

```

```

//remove any leading or trailing whitespace from input
input.trim();

//convert string to uppercase if needed
input.toUpperCase();

//goes through the for loop to use the buzzer and LED
for(int i = 0; i < input.length(); i++)
{
    //checks if the current character is a letter
    char currentChar = input[i];

    if (isalpha(currentChar))
    {
        //get Morse code
        //const char* morse = getMorseCode(currentChar);

        //blink Morse code
        blinkMorseCode(currentChar);
    }
    //if it's not a letter
    else
    {
        delay(1000);
        Serial.println("Invalid character entered!");
    }
}

//wait before repeating
delay(1400);
Serial.println("Enter a new string: ");
}
}

```

Logbook from date to date

Day	Logbook of work
Wednesday, 18.09.2024	I started thinking about how I can approach this project and what to begin with.
Thursday, 19.09.2024	I started with the easier parts of the assignment. The first thing I

	did was to create the two arrays and void setup(). After that, I defined the LED and buzzer, then made them execute actions at the same time.
Friday, 20.09.2024	I worked on the void loop().
Saturday, 21.09.2024	I struggled with the outputs of the letters and their specific Morse code, but I managed to get over it.

Week 2 - Self reflection

For the Embedded Systems class we had to make a program that transformed words into Morse Code. This assignment was a bit challenging, because we had to light up an LED and make the buzzer sound at the same time. In addition, I was happy when I finished this project and showed it to my sister. She was amazed that I managed to do something like this.

For Computer Science class we had to do three assignments: Scrabble, Readability, Substitution. Thanks to these assignments, I understood how arrays are used in C programming language. I enjoyed the “Substitution” project most, because I had to encrypt a message by replacing every letter with another letter.

In my opinion, the Dutch class is the fun part of the week. We learn new words and struggle to pronounce them like the Dutch people.

Week 3 – Algorithms

Sort

- Selection sort iterates through the unsorted portions of a list, selecting the smallest element each time and moving it to its correct location.
- Bubble sort compares pairs of adjacent values one at a time and swaps them if they are in the incorrect order. This continues until the list is sorted.
- Merge sort recursively divides the list into two repeatedly and then merges the smaller lists back into a larger one in the correct order.

In this problem, you'll analyze three (compiled!) sorting programs to determine which algorithms they use. In a file called `answers.txt` in a folder called `sort`, record your answers, along with an explanation for each program, by filling in the blanks marked `TODO`.

```
answers.txt X
1 sort1 uses: Bubble Sort
2
3 How do you know?: Sort 1 was the second one to finish checking the sorted lists.
4
5 sort2 uses: Merge Sort
6
7 How do you know?: Sort 2 was the fastest one to arrange the random lists of numbers.
8
9 sort3 uses: Selection Sort
10
11 How do you know?: Sort 3 was the last one to check every list of numbers.
12
```

Logbook from date to date

Day	Logbook of work
Tuesday, 24.09.2024	Done the assignment.

The Harvard score with the title of the program and my name:

[me50](#) / [users](#) / [Alexia220700](#) / [cs50](#) / [problems](#) / [2024](#) / [x](#) / [sort](#)

🔑 #1 submitted a few seconds ago, Tuesday, September 24, 2024 8:12 PM CEST
[check50](#) [3/3](#) • [0 comments](#)
[tar.gz](#) • [zip](#)

<https://submit.cs50.io/check50/9de62ada7e3bb9a23ced6332935ee7d281dec90b>

Plurality

Elections come in all shapes and sizes. In the UK, the Prime Minister is officially appointed by the monarch, who generally chooses the leader of the political party that wins the most seats in the House of Commons. The United States uses a multi-step Electoral College process where citizens vote on how each state should allocate Electors who then elect the President.

Perhaps the simplest way to hold an election, though, is via a method commonly known as the “plurality vote” (also known as “first-past-the-post” or “winner take all”). In the plurality vote, every voter gets to vote for one candidate. At the end of the election, whichever candidate has the greatest number of votes is declared the winner of the election.

For this problem, you’ll implement a program that runs a plurality election, per the below.

This is the code for the assignment:

```
#include <cs50.h>
#include <stdio.h>
#include <string.h>

// Max number of candidates
#define MAX 9

// Candidates have name and vote count
typedef struct
{
    string name;
    int votes;
} candidate;

// Array of candidates
candidate candidates[MAX];

// Number of candidates
int candidate_count;
```

```

// Function prototypes
bool vote(string name);
void print_winner(void);

int main(int argc, string argv[])
{
    // Check for invalid usage
    if (argc < 2)
    {
        printf("Usage: plurality [candidate ...]\n");
        return 1;
    }

    // Populate array of candidates
    candidate_count = argc - 1;
    if (candidate_count > MAX)
    {
        printf("Maximum number of candidates is %i\n", MAX);
        return 2;
    }
    for (int i = 0; i < candidate_count; i++)
    {
        candidates[i].name = argv[i + 1];
        candidates[i].votes = 0;
    }

    int voter_count = get_int("Number of voters: ");

    // Loop over all voters
    for (int i = 0; i < voter_count; i++)
    {
        string name = get_string("Vote: ");

        // Check for invalid vote
        if (!vote(name))
        {
            printf("Invalid vote.\n");
        }
    }

    // Display winner of election
    print_winner();
}

// Update vote totals given a new vote
bool vote(string name)
{

```

```

// go through the array and check if the input name can be found in it
for (int i = 0; i <= candidate_count - 1; i++)
{
    if (strcmp(candidates[i].name, name) == 0)
    {
        // if yes, add the vote to that candidate
        candidates[i].votes++;
        return true;
    }
}
return false;
}

// Print the winner (or winners) of the election
void print_winner(void)
{
    int max = 0;

    // Selection Sort, first loop
    for (int i = 0; i <= candidate_count - 1; i++)
    {
        // if a candidate has more votes than max
        if (candidates[i].votes > max)
        {
            // max = number of votes of that candidate
            max = candidates[i].votes;
        }
    }
    // second loop to find outt if there are two winners or not
    for (int j = 0; j <= candidate_count - 1; j++)
    {
        // if there are two winners, we have to remember their name in a string
        if (candidates[j].votes == max)
        {
            printf("%s\n", candidates[j].name);
        }
    }
}

```

Logbook from date to date

Day	Logbook of work
Tuesday, 24.09.2024	Done the assignment.

The Harvard score with the title of the program and my name:

[me50](#) / [users](#) / [Alexia220700](#) / [cs50](#) / [problems](#) / [2024](#) / [x](#) / [plurality](#)

🔑 #1 submitted a few seconds ago, Tuesday, September 24, 2024 10:01 PM CEST
[check50](#) 14/14 • [style50](#) 1.00 • 0 comments
[tar.gz](#) • [zip](#)

<https://submit.cs50.io/check50/e7ddfa8f56a47d30a40b59dbb001e763ab07ccb7>

Runoff

There's another kind of voting system known as a ranked choice voting system. In a ranked choice system, voters can vote for more than one candidate. Instead of just voting for their top choice, they can rank the candidates in order of preference.

If any candidate has a majority (more than 50%) of the first preference votes, that candidate is declared the winner of the election.

If no candidate has more than 50% of the vote, then an “instant runoff” occurs. The candidate who received the fewest number of votes is eliminated from the election, and anyone who originally chose that candidate as their first preference now has their second preference considered.

Once a candidate has a majority, that candidate is declared the winner.

This is the code for the assignment:

```
#include <cs50.h>
#include <stdio.h>
#include <string.h>

// Max voters and candidates
#define MAX_VOTERS 100
#define MAX_CANDIDATES 9

// preferences[i][j] is jth preference for voter i
int preferences[MAX_VOTERS][MAX_CANDIDATES];

// Candidates have name, vote count, eliminated status
typedef struct
{
    string name;
    int votes;
    bool eliminated;
} candidate;

// Array of candidates
candidate candidates[MAX_CANDIDATES];
```

```

// Numbers of voters and candidates
int voter_count;
int candidate_count;

// Function prototypes
bool vote(int voter, int rank, string name);
void tabulate(void);
bool print_winner(void);
int find_min(void);
bool is_tie(int min);
void eliminate(int min);

int main(int argc, string argv[])
{
    // Check for invalid usage
    if (argc < 2)
    {
        printf("Usage: runoff [candidate ...]\n");
        return 1;
    }

    // Populate array of candidates
    candidate_count = argc - 1;
    if (candidate_count > MAX_CANDIDATES)
    {
        printf("Maximum number of candidates is %i\n", MAX_CANDIDATES);
        return 2;
    }
    for (int i = 0; i < candidate_count; i++)
    {
        candidates[i].name = argv[i + 1];
        candidates[i].votes = 0;
        candidates[i].eliminated = false;
    }

    voter_count = get_int("Number of voters: "); // ask for nr of voters
    if (voter_count > MAX_VOTERS)                // if an uncertain number is entered
    {
        printf("Maximum number of voters is %i\n", MAX_VOTERS);
        return 3;
    }

    // Keep querying for votes                                //asks for votes *rank from
voters
    // voter_count
    for (int i = 0; i < voter_count; i++)
    {

```

```

// Query for each rank
for (int j = 0; j < candidate_count; j++) // by rank 1 2 3... candidate_count
{
    string name = get_string("Rank %i: ", j + 1);

    // Record vote, unless it's invalid
    if (!vote(i, j, name))
    {
        printf("Invalid vote.\n");
        return 4;
    }
}

printf("\n");
}

// Keep holding runoffs until winner exists //try to find winner
while (true)
{
    // Calculate votes given remaining candidates
    tabulate();

    // Check if election has been won
    bool won = print_winner();
    if (won)
    {
        break;
    }

    // Eliminate last-place candidates //last place candidate is eliminated
    int min = find_min();
    bool tie = is_tie(min);

    // If tie, everyone wins
    if (tie)
    {
        for (int i = 0; i < candidate_count; i++)
        {
            if (!candidates[i].eliminated)
            {
                printf("%s\n", candidates[i].name);
            }
        }
        break;
    }
}

```

```

        // Eliminate anyone with minimum number of votes
        eliminate(min);

        // Reset vote counts back to zero
        for (int i = 0; i < candidate_count; i++)
        {
            candidates[i].votes = 0;
        }
    }
    return 0;
}

// Record preference if vote is valid
bool vote(int voter, int rank, string name)
{
    // check if candidate's name exists in the array
    for (int i = 0; i <= candidate_count - 1; i++)
    {
        // checking if the candidate is valid
        if (strcmp(candidates[i].name, name) == 0)
        {
            // if the candidate's name is valid, record preference
            preferences[voter][rank] = i;
            return true;
        }
    }
    return false;
}

// Tabulate votes for non-eliminated candidates
void tabulate(void)
{
    // first loop goes through every voter
    for (int i = 0; i <= voter_count - 1; i++)
    {
        // second loop goes through the candidates
        for (int j = 0; j < candidate_count; j++)
        {
            //int the vote for a specific candidate
            int candidate_index = preferences[i][j];

            // verify if the candidate is eliminated or not
            if (!candidates[candidate_index].eliminated)
            {
                // if candidate is not eliminated, the vote is added to him
                candidates[candidate_index].votes++;
                break;
            }
        }
    }
}

```

```

    }
}
return;
}

// Print the winner of the election, if there is one
bool print_winner(void)
{
    //loop through the candidates
    for (int i = 0; i <= candidate_count - 1; i++)
    {
        //if the candidate has more votes than half of the votes, he is the winner
        if (candidates[i].votes > voter_count / 2)
        {
            printf("%s\n", candidates[i].name);
            return true;
        }
    }

    return false;
}

// Return the minimum number of votes any remaining candidate has
int find_min(void)
{
    int min = voter_count;
    //loop through candidates
    for (int i = 0; i <= candidate_count - 1; i++)
    {
        //if the candidate is not eliminated
        if (!candidates[i].eliminated)
        {
            //if votes are less than minimum, his number of votes becomes the new min
            if (candidates[i].votes < min)
            {
                min = candidates[i].votes;
            }
        }
    }

    return min;
}

// Return true if the election is tied between all candidates, false otherwise
bool is_tie(int min)
{

```

```

// int sum used for checking if all of the candidates have equal votes
int sum = 0;
int sum2 = 0;

// loop through candidates array
for (int i = 0; i < candidate_count; i++)
{
    // counting the non-eliminated candidates
    if (!candidates[i].eliminated)
    {
        // number of candidates
        sum++;

        // compare number of votes to min
        if (candidates[i].votes == min)
        {
            // if they are equal, count them
            sum2++;
        }
    }
}

// compare the number of candidates with the number of
// candidates with minimum votes
if (sum == sum2)
{
    return true;
}
else
{
    return false;
}

return false;
}

// Eliminate the candidate (or candidates) in last place
void eliminate(int min)
{
    // loop through the candidates
    for (int i = 0; i <= candidate_count - 1; i++)
    {
        //if the candidate got the min votes, he is eliminated
        if (candidates[i].votes == min)
        {
            candidates[i].eliminated = true;
        }
    }
}

```

```

}

return;
}

```

Logbook from date to date

Day	Logbook of work
Wednesday, 25.09.2024	I completed the vote function and the print_winner function.
Thursday, 26.09.2024	I completed tabulate function and find minimum function. After that, I checked the code on CS50.

The Harvard score with the title of the program and my name:

[me50](#) / [users](#) / [Alexia220700](#) / [cs50](#) / [problems](#) / [2024](#) / [x](#) / [runoff](#)

🔑 #1 submitted a few seconds ago, Thursday, September 26, 2024 2:51 PM CEST
[check50](#) 25/25 • [style50](#) 1.00 • 0 comments
[tar.gz](#) • [zip](#)

<https://submit.cs50.io/check50/11b3ecbc26db2792db9b385b8f5586768939949b>

Week 3 - Traffic light project with pedestrian crossing

In this project, a program is written to control traffic lights at a simple junction. Red, yellow and green LEDs are used to simulate traffic lights.

Pressing the pedestrian button stops the traffic on both roads by setting both lights to red for 15 seconds. Access to the pedestrian crossing is only given at the end of a cycle when the lights on both roads are red. If the lights aren't red, then the pedestrian must wait until the cycle is finished and both lights turn red.

This is the code for the assignment:

```
//defines pedestrian pushbutton, input
#define pedpb 2

//outputs
//latch pin
int STCP = 5;
//clock pin
int SHCP = 6;
//data pin
int DS = 4;

//states
//pedestrianButton checks if button is pushed or not, if yes, pedestrianButton = 1
bool pedestrianButton = 0;
//checks if pedestrians are in process of crossing, if yes, pedestrianCrossing = 1
bool pedestrianCrossing = 0;

//traffic light states
byte trafficLightStates[] =
{
    //north sequence(while east is red)
    B01001001, //north red, east red, pedestrian red
    B01001011, //north red + yellow, east red, pedestrian red
    B01001100, //north green, east red, pedestrian red
    B01001010, //north yellow, east red, pedestrian red
```



```

//east sequence(while north is red)
B01001001, //north red, east red, pedestrian Red
B01011001, //north red, east red + yellow, pedestrian red
B01100001, //north red, east green, pedestrian red
B01010001 //north red, east yellow, pedestrian red
};

//timing for each state of traffic lights in milliseconds
int stateDelays[] =
{
    //red for both or for east
    3000, //red for both
    2000, //red + yellow for north, red for east
    5000, //green for north, red for east
    2000, //yellow for North, red for east
    3000, //red for both

    //red for both or for north
    3000, //red for both
    2000, //red for north, red + yellow for east
    5000, //red for north, green for east
    2000, //red for north, yellow for east
    3000 //red for both
};

//pedestrian state
//pedestrian green, north and east red
byte pedestrianState = B10001001;

//pedestrian green for 15 seconds while north and east are red
int pedDelay = 15000;

void setup()
{
    //latch, output
    pinMode(STCP, OUTPUT);
    //clock, output
    pinMode(SHCP, OUTPUT);
    //data, output
    pinMode(DS, OUTPUT);

    //pedestrian button input
    pinMode(pedpb, INPUT_PULLUP);
    //interrupt on push button when pressed
    attachInterrupt(digitalPinToInterrupt(pedpb), PedRequest, FALLING);

```

```

}

void PedRequest()
{
    //when button is pushed
    pedestrianButton = 1;
}

void loop()
{
    //traffic light sequence
    for (int i = 0; i < sizeof(trafficLightStates) / sizeof(trafficLightStates[0]); i++)
    {
        //if the pedestrian button is pressed, interrupt the normal cycle of traffic lights
        if ((trafficLightStates[i] & B01001001) == B01001001 && pedestrianButton
&& !pedestrianCrossing)
        {
            //delay 1000 milliseconds
            delay(1000);

            //let pedestrians cross the street
            pedestrianCrossing = 1;

            //reset the request
            pedestrianButton = 0;

            //pedestrian crossing state
            digitalWrite(STCP, LOW); // Latch LOW

            //sends the pedestrianState byte to a shift register
            shiftOut(DS, SHCP, LSBFIRST, pedestrianState);

            //latch high
            digitalWrite(STCP, HIGH);

            //wait for pedestrian crossing delay
            delay(pedDelay);

            //reset pedestrian crossing state
            pedestrianCrossing = 0;
        }

        //continue normal traffic light sequence
        //latch low
        digitalWrite(STCP, LOW);
    }
}

```

```

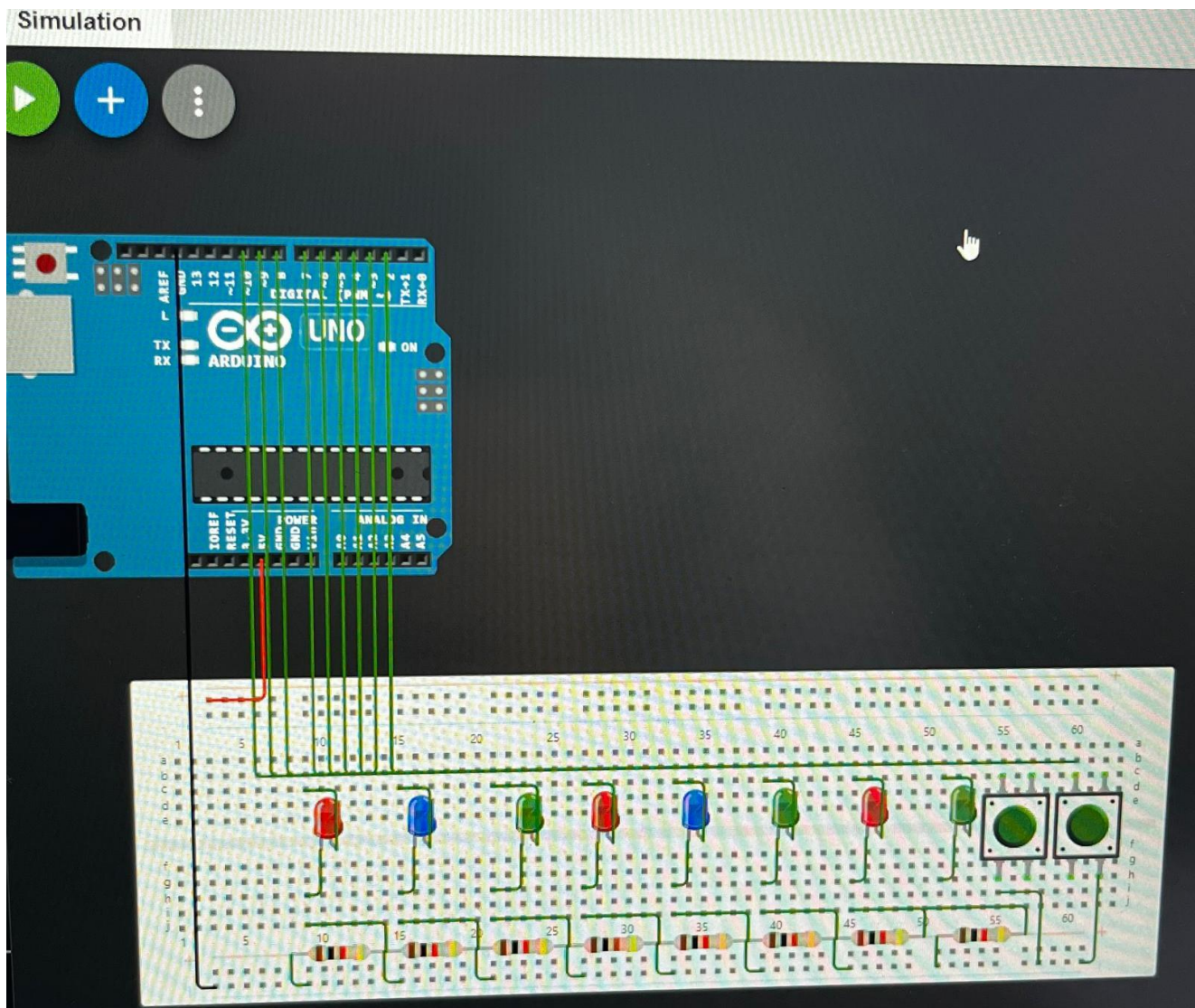
//shift out the current light state
shiftOut(DS, SHCP, LSBFIRST, trafficLightStates[i]);

//latch high
digitalWrite(STCP, HIGH);

//wait for delay
delay(stateDelays[i]);
}
}

```

This is a picture of my Traffic Lights assignment, made on the online Arduino:



Logbook from date to date

Day	Logbook of work
Wednesday, 25.09.2024	I started working on the project on online Arduino. I made code and three LEDs work.
Thursday, 26.09.2024	I made all the LEDs work on the online Arduino.
Friday, 27.09.2024	I added the buttons on the online Arduino.
Saturday, 28.09.2024	I started to work on a real-life project. The LEDs worked, but the buttons didn't.
Monday, 30.09.2024	The buttons worked, but the LEDs didn't.
Tuesday, 01.10.2024	I kept trying to fix the project.
Wednesday, 02.10.2024	The project worked, but one of the LEDs was lighting less than the rest.
Friday, 04.10.2024	I changed again the LEDs, and the project worked perfectly.

Week 3 - Self reflection

Computer Science class taught me about two different voting systems. In the first one, named “plurality vote” or “winner takes all”, every voter gets to vote for one candidate. At the end of the election, whichever candidate has the greatest number of votes is declared the winner of the election. On the other hand, the second one, intitulated “Runoff”, a ranked-choice system, voters can vote for more than one candidate. Instead of just voting for their top choice, they can rank the candidates in order of preference. In my opinion, these assignments are important to understand how different types of voting systems work around the World.

Embedded Systems class was a challenge for me, because we had to do the “Traffic lights with pedestrian crossing” assignment. Firstly, I made the code and the scheme for the Breadboard on an online Arduino. It worked perfectly so I thought it would work like that in real life too. To my surprise, when I tried to do it on my Breadboard, my LEDs were working, but the buttons didn’t. When I made the buttons work, the LEDs stopped lighting up. After two weeks, I managed to fix this complex project and present it.

Week 4 – Memory

Volume

Complete the implementation of `volume.c`, such that it changes the volume of a sound file by a given factor.

The program should accept three command-line arguments. The first is input, which represents the name of the original audio file. The second is output, which represents the name of the new audio file that should be generated. The third factor, which is the amount by which the volume of the original audio file should be scaled.

For example, if factor is 2.0, then your program should double the volume of the audio file in input and save the newly generated audio file in output.

Your program should first read the header from the input file and write the header to the output file.

Your program should then read the rest of the data from the WAV file, one 16-bit (2-byte) sample at a time. Your program should multiply each sample by the factor and write the new sample to the output file.

Your program, if it uses `malloc`, must not leak any memory.

This is the code for the assignment:

```
// Modifies the volume of an audio file

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

// Number of bytes in .wav header
const int HEADER_SIZE = 44;

// argc ARGument Count == int variable that stores the number of command-line arguments
```

```

// argv ARGument Vector == array of character pointers listing
int main(int argc, char *argv[])
{
    // Check command-line arguments
    if (argc != 4)
    {
        printf("Usage: ./volume input.wav output.wav factor\n");
        return 1;
    }

    // Open files and determine scaling factor
    FILE *input = fopen(argv[1], "r");
    if (input == NULL)
    {
        printf("Could not open file.\n");
        return 1;
    }

    FILE *output = fopen(argv[2], "w");
    if (output == NULL)
    {
        printf("Could not open file.\n");
        return 1;
    }

    // atof used to convert a string into a floating-point number
    // and represent the converted floating point nr to its corresponding
    // double value
    float factor = atof(argv[3]);

    // copy header from input file to output file

    // array of bytes
    // uint8_t is a type that stores an 8-bit integer
    // n = HEADER_SIZE, which is equal to 44
    uint8_t header[HEADER_SIZE];

    // reads bytes from a file
    fread(header, HEADER_SIZE, 1, input);

    // writes bytes to a file
    fwrite(header, HEADER_SIZE, 1, output);

    // read samples from input file and write updated data to output file
    // Create a buffer for a single sample
    int16_t buffer;

```

```

// Read single sample from input into buffer while there are samples left to read
// using fread() function
while (fread(&buffer, sizeof(int16_t), 1, input))
{
    // Update volume of sample
    buffer *= factor;

    // Write updated sample to new file using fwrite()
    fwrite(&buffer, sizeof(int16_t), 1, output);
}

// Close files
fclose(input);
fclose(output);
}

```

Logbook from date to date

Day	Logbook of work
Wednesday, 02.10.2024	Done the assignment.

The Harvard score with the title of the program and my name:

[me50](#) / [users](#) / [Alexia220700](#) / [cs50](#) / [problems](#) / [2024](#) / [x](#) / [volume](#)

🔗 #1 submitted 12 hours ago, Wednesday, October 2, 2024 9:56 PM CEST
 check50 5/5 • style50 1.00 • 0 comments
[tar.gz](#) • [zip](#)

<https://submit.cs50.io/check50/069e2361f466bebf389b37d2f89f70c72e092e4f>

Filter-less

Implement the functions in helpers.c such that a user can apply grayscale, sepia, reflection, or blur filters to their images.

The function grayscale should take an image and turn it into a black-and-white version of the same image.

The function sepia should take an image and turn it into a sepia version of the same image.

The reflect function should take an image and reflect it horizontally.

Finally, the blur function should take an image and turn it into a box-blurred version of the same image.

You should not modify any of the function signatures, nor should you modify any other files other than helpers.c.

This is the code for the assignment:

```
#include "helpers.h"
#include <math.h>

// Convert image to grayscale
void grayscale(int height, int width, RGBTRIPLE image[height][width])
{
    for (int i = 0; i < height; i++)
    {
        for (int j = 0; j < width; j++)
        {
            int avg;
            // calculate the average pixel value
            // round the number to make sure that the average is correctly rounded
            // to the nearest integer
            // division done in floating-point arithmetic
            avg = round((image[i][j].rgbtRed + image[i][j].rgbtGreen +
image[i][j].rgbtBlue) / 3.0);
            // set each color value to the average value
            image[i][j].rgbtRed = avg;
            image[i][j].rgbtGreen = avg;
            image[i][j].rgbtBlue = avg;
        }
    }
}
```

```

    }
    return;
}

// Convert image to sepia
void sepia(int height, int width, RGBTRIPLE image[height][width])
{
    // int sepiaRed;
    // int sepiaGreen;
    // int sepiaBlue;

    // for each pixel
    for (int i = 0; i < height; i++)
    {
        for (int j = 0; j < width; j++)
        {
            // calculate each new color value using the sepia formula
            float sepiaRed = 0.393 * image[i][j].rgbtRed + 0.769 * image[i][j].rgbtGreen +
                0.189 * image[i][j].rgbtBlue;
            float sepiaGreen = 0.349 * image[i][j].rgbtRed + 0.686 * image[i][j].rgbtGreen
+
                0.168 * image[i][j].rgbtBlue;
            float sepiaBlue = 0.272 * image[i][j].rgbtRed + 0.534 * image[i][j].rgbtGreen
+
                0.131 * image[i][j].rgbtBlue;

            // ensure the result is an integer between 0 and 255, inclusive
            if (sepiaRed > 255)
            {
                image[i][j].rgbtRed = 255;
            }
            else
            {
                // round the number
                image[i][j].rgbtRed = round(sepiaRed);
            }

            // assign values
            // same as the sepiaRed, but written different
            // round the number
            image[i][j].rgbtGreen = sepiaGreen > 255 ? 255 : round(sepiaGreen);
            image[i][j].rgbtBlue = sepiaBlue > 255 ? 255 : round(sepiaBlue);
        }
    }

    return;
}

```

```

// Reflect image horizontally
void reflect(int height, int width, RGBTRIPLE image[height][width])
{
    // first loop is used for rows
    for (int i = 0; i < height; i++)
    {
        // second loop is used to swap the pixels
        for (int j = 0; j < width / 2; j++)
        {
            // temporary variable that stores pixels on certain positions
            RGBTRIPLE aux = image[i][j];

            // swap the pixel at j with the pixel at width - j - 1
            image[i][j] = image[i][width - j - 1];
            image[i][width - j - 1] = aux;
        }
    }
    return;
}

// Blur image
void blur(int height, int width, RGBTRIPLE image[height][width])
{
    // create a copy of the image so it doesn't affect the other pixels
    RGBTRIPLE copy[height][width];
    for (int i = 0; i < height; i++)
    {
        for (int j = 0; j < width; j++)
        {
            copy[i][j] = image[i][j];
        }
    }

    // loop through the pixels in the image
    for (int i = 0; i < height; i++)
    {
        for (int j = 0; j < width; j++)
        {
            int redSum = 0;
            int greenSum = 0;
            int blueSum = 0;
            // avg is a counter
            int counter = 0;

            // loop over the neighbouring pixels
            // by creating the 3x3 grid

```

```

        for (int x = -1; x <= 1; x++)
        {
            for (int y = -1; y <= 1; y++)
            {
                // checking for inbound
                if ((i + x) >= 0 && (i + x) < height && (j + y) >= 0 && (j + y) <
width)
                {
                    // if it's a valid neighbour, sum its colors
                    redSum += copy[i + x][j + y].rgbtRed;
                    greenSum += copy[i + x][j + y].rgbtGreen;
                    blueSum += copy[i + x][j + y].rgbtBlue;
                    // count the valid neighbours
                    counter++;
                }
            }
        }

        // assign the average values to the current pixel
        // round them to make sure that the averages are correctly rounded
        // to the nearest integer
        image[i][j].rgbtRed = round((float) redSum / counter);
        image[i][j].rgbtGreen = round((float) greenSum / counter);
        image[i][j].rgbtBlue = round((float) blueSum / counter);
    }
}

return;
}

```

Logbook from date to date

Day	Logbook of work
Monday, 30.09.2024	Started the project. I did the void greyscale and the void sepia.
Wednesday, 02.10.2024	I did the void reflect
Thursday, 03.10.2024	I did the void blur. Done the assignment.

The Harvard score with the title of the program and my name:

[me50](#) / [users](#) / [Alexia220700](#) / [cs50](#) / [problems](#) / [2024](#) / [x](#) / [filter](#) / [less](#)

[My Submissions](#)

[My Courses](#)

[Docs](#)

[Log Out](#)

🔑 #1 submitted a minute ago, Thursday, October 3, 2024 11:09 AM CEST

[check50](#) 22/22 • [style50](#) 1.00 • 0 comments

[tar.gz](#) • [zip](#)

<https://submit.cs50.io/check50/531930d76512bfa624cc7804f8ee39616f6d47d6>

Filter-more

Implement the functions in helpers.c such that a user can apply grayscale, reflection, blur, or edge detection filters to their images.

The function grayscale should take an image and turn it into a black-and-white version of the same image.

The reflect function should take an image and reflect it horizontally.

The blur function should take an image and turn it into a box-blurred version of the same image.

The edges function should take an image and highlight the edges between objects, according to the Sobel operator.

You should not modify any of the function signatures, nor should you modify any other files other than helpers.c.

This is the code for the assignment:

```
#include "helpers.h"
#include <math.h>
#include <stdint.h> //used for rounding

// Convert image to grayscale
void grayscale(int height, int width, RGBTRIPLE image[height][width])
{
    for (int i = 0; i < height; i++)
    {
        for (int j = 0; j < width; j++)
        {
            int avg;
            // calculate the average pixel value
            // round the number to make sure that the average is correctly rounded
            // to the nearest integer
            // division done in floating-point arithmetic
            avg = round((image[i][j].rgbtRed + image[i][j].rgbtGreen +
image[i][j].rgbtBlue) / 3.0);
            // set each color value to the average value
            image[i][j].rgbtRed = avg;
            image[i][j].rgbtGreen = avg;
```

```

        image[i][j].rgbtBlue = avg;
    }
}
return;
}

// Reflect image horizontally
void reflect(int height, int width, RGBTRIPLE image[height][width])
{
    // first loop is used for rows
    for (int i = 0; i < height; i++)
    {
        // second loop is used to swap the pixels
        for (int j = 0; j < width / 2; j++)
        {
            // temporary variable that stores pixels on certain positions
            RGBTRIPLE aux = image[i][j];

            // swap the pixel at j with the pixel at width - j - 1
            image[i][j] = image[i][width - j - 1];
            image[i][width - j - 1] = aux;
        }
    }
    return;
}

// Blur image
void blur(int height, int width, RGBTRIPLE image[height][width])
{
    // create a copy of the image so it doesn't affect the other pixels
    // RGBTRIPLE means uint_8 rgbtRed; uint_8 rgbtBlue; uint_8 rgbtGreen;
    RGBTRIPLE copy[height][width];

    // Copy the original image into the copy array
    for (int i = 0; i < height; i++)
    {
        for (int j = 0; j < width; j++)
        {
            copy[i][j] = image[i][j];
        }
    }

    for (int i = 0; i < height; i++)
    {
        for (int j = 0; j < width; j++)
        {
            int redSum = 0;

```

```

    int greenSum = 0;
    int blueSum = 0;

    // used for counting neighbouring pixels
    int counter = 0;

    // loop over the neighbouring pixels
    // by creating the 3x3 grid
    for (int x = -1; x <= 1; x++)
    {
        for (int y = -1; y <= 1; y++)
        {
            // checking for valid neighbouring pixels
            if ((i + x) >= 0 && (i + x) < height && (j + y) >= 0 && (j + y) <
width)
            {
                // if it's a valid neighbour, sum its colors
                redSum += copy[i + x][j + y].rgbtRed;
                greenSum += copy[i + x][j + y].rgbtGreen;
                blueSum += copy[i + x][j + y].rgbtBlue;

                // count the valid neighbours
                counter++;
            }
        }
    }

    // assign the average values to the current pixel
    // round them to make sure that the averages are correct
    image[i][j].rgbtRed = round((float) redSum / counter);
    image[i][j].rgbtGreen = round((float) greenSum / counter);
    image[i][j].rgbtBlue = round((float) blueSum / counter);
}
}

return;
}

// Detect edges
void edges(int height, int width, RGBTRIPLE image[height][width])
{
    // create a copy of the image so it doesn't affect the other pixels
    // RGBTRIPLE means uint_8 rgbtRed; uint_8 rgbtBlue; uint_8 rgbtGreen;
    RGBTRIPLE copy[height][width];
    for (int i = 0; i < height; i++)
    {
        for (int j = 0; j < width; j++)

```



```

    {
        copy[i][j] = image[i][j];
    }
}

// loop through the pixels in the image
for (int i = 0; i < height; i++)
{
    for (int j = 0; j < width; j++)
    {
        // initializing each gradient sum
        int gxRed = 0;
        int gxGreen = 0;
        int gxBlue = 0;
        int gyRed = 0;
        int gyBlue = 0;
        int gyGreen = 0;

        // making two arrays
        int gx[9] = {-1, 0, 1, -2, 0, 2, -1, 0, 1};
        int gy[9] = {1, 2, 1, 0, 0, 0, -1, -2, -1};

        // counter is used for checking if there are any neighbouring pixels
        // reset counter for each pixel
        int counter = 0;

        // looping over the neighbouring pixels
        // by creating the 3x3 grid
        for (int x = -1; x <= 1; x++)
        {
            for (int y = -1; y <= 1; y++)
            {
                // checking for valid neighbouring pixels, in bounds
                if ((i + x) >= 0 && (i + x) < height && (j + y) >= 0 && (j + y) <
width)
                {
                    // making copies
                    gxRed += copy[i + x][j + y].rgbtRed * gx[counter];
                    gxGreen += copy[i + x][j + y].rgbtGreen * gx[counter];
                    gxBlue += copy[i + x][j + y].rgbtBlue * gx[counter];

                    gyRed += copy[i + x][j + y].rgbtRed * gy[counter];
                    gyGreen += copy[i + x][j + y].rgbtGreen * gy[counter];
                    gyBlue += copy[i + x][j + y].rgbtBlue * gy[counter];
                }
                // counter for valid member
                counter++;
            }
        }
    }
}

```

```

    }
}
// calculate the final gradient magnitude for each channel
int redValue = round(sqrt(gxRed * gxRed + gyRed * gyRed));
int greenValue = round(sqrt(gxGreen * gxGreen + gyGreen * gyGreen));
int blueValue = round(sqrt(gxBlue * gxBlue + gyBlue * gyBlue));

// claim the values from 0 to 255
// if value is more than 255, then value turns into 255
if (redValue > 255)
{
    redValue = 255;
}
if (greenValue > 255)
{
    greenValue = 255;
}
if (blueValue > 255)
{
    blueValue = 255;
}

// assign new values to image
image[i][j].rgbtRed = redValue;
image[i][j].rgbtGreen = greenValue;
image[i][j].rgbtBlue = blueValue;
}
}
// return after all pixels are processed
return;
}

```

Logbook from date to date

Day	Logbook of work
Wednesday, 02.10.2024	Started the assignment.
Thursday, 03.10.2024	Done the assignment.

The Harvard score with the title of the program and my name:

[me50](#) / [users](#) / [Alexia220700](#) / [cs50](#) / [problems](#) / [2024](#) / [x](#) / [filter](#) / [more](#)

🔗 #1 submitted a few seconds ago, Thursday, October 3, 2024 10:35 PM CEST

[check50](#) 23/23 • [style50](#) 1.00 • 0 comments

[tar.gz](#) • [zip](#)

<https://submit.cs50.io/check50/92e7edb680884adb9b03795c691e6690f9bae59e>

Recover

Implement a program called `recover` that recovers JPEGs from a forensic image.

Implement your program in a file called `recover.c` in a directory called `recover`.

Your program should accept exactly one command-line argument, the name of a forensic image from which to recover JPEGs.

If your program is not executed with exactly one command-line argument, it should remind the user of correct usage, and `main` should return 1.

If the forensic image cannot be opened for reading, your program should inform the user as much, and `main` should return 1.

The files you generate should each be named `###.jpg`, where `###` is a three-digit decimal number, starting with 000 for the first image and counting.

Your program, if it uses `malloc`, must not leak any memory.

This is the code for the assignment:

```
#include <stdio.h>
#include <stdlib.h>
// for uint8_t
#include <stdint.h>

// one command line argument

int main(int argc, char *argv[])
{
    // accept a single command-line argument
    if (argc != 2)
    {
        printf("Usage: ./recover FILE\n");
        return 1;
    }

    // opens the memory card in read mode only
```

```

FILE *f = fopen(argv[1], "r");

// check if memory card opens correctly
if (f == NULL)
{
    // printf("Could not open file %s.\n", argv[1]);
    return 2;
}

// create a buffer
uint8_t buffer[512];

// create a buffer to store all the filenames
// size of file = 8 bytes
char filename[8];

// counts every JPEG file
int counter = 0;

FILE *img = NULL;
// while there's still data left to read from the memory card
// fread(data, size, number, inptr)
// fread checks in 512 byte blocks
while (fread(buffer, 1, 512, f) == 512)
{
    // checks if there are any JPEG headers in the array / buffer
    // the buffer[3] part clears out the last 4 bits by equaling them to 0 and only
checks the
    // first 4 bits
    if (buffer[0] == 0xff && buffer[1] == 0xd8 && buffer[2] == 0xff &&
        ((buffer[3] & 0xf0) == 0xe0))
    {
        // closes previous file if it exists
        if (img != NULL)
        {
            fclose(img);
        }

        // creates a new file with a sequential name
        sprintf(filename, "%03i.jpg", counter);
        // opens file in write mode
        img = fopen(filename, "w");

        if (img == NULL) // Check if the JPEG file was created successfully
        {
            fclose(f); // Close the memory card file before exiting

```

```

        return 1;
    }

    // writes the new buffer in the new file
    fwrite(buffer, 1, 512, img);

    // counts each JPEG header
    counter++;
}
// if there is no new header
else if (img != NULL)
{
    // continues to write in this buffer
    fwrite(buffer, 1, 512, img);
}
}

// closes previous file if it exists
if (img != NULL)
{
    fclose(img);
}

// close the memory card
fclose(f);
return 0;
}

```

Logbook from date to date

Day	Logbook of work
Friday, 04.10.2024	Started the assignment.
Saturday, 05.10.2024	Done the assignment.

The Harvard score with the title of the program and my name:

[me50](#) / [users](#) / [Alexia220700](#) / [cs50](#) / [problems](#) / [2024](#) / [x](#) / [recover](#)

🔑 #1 submitted a few seconds ago, Saturday, October 5, 2024 11:46 PM CEST
check50 7/7 • style50 1.00 • 0 comments
[tar.gz](#) • [zip](#)

<https://submit.cs50.io/check50/0b73ad95e57a268fd62e666892b427ce851962e0>

Week 4 – Displaying Uptime

This project demonstrates a digital clock implementation using a 4-digit, 7-segment display to show minutes and seconds, incrementing each second up to a maximum of 99 minutes and 59 seconds. The display is controlled via Arduino, with each digit displayed one at a time using multiplexing.

This is the code for the assignment:

```
//define for segments
#define ON HIGH
#define OFF LOW

//define for digits
#define UP HIGH
#define DOWN LOW

//port number for each segment
int LED[] =
{
    2,  //a
    3,  //b
    4,  //c
    5,  //d
    6,  //e
    7,  //f
    8   //g
};

//port number for each digit
unsigned char DIGITS[] =
{
    9,    //DIG1
    10,   //DIG2
    11,   //DIG3
    12    //DIG4
};

//states of each segment of the number
byte segmentStates[] =
{
    B00111111,  //0
```



```

    B00000110,    //1
    B01011011,    //2
    B01001111,    //3
    B01100110,    //4
    B01101101,    //5
    B01111101,    //6
    B00000111,    //7
    B01111111,    //8
    B01101111     //9
};

int minutes = 0;
int seconds = 0;

//starts the counting from 0
unsigned long previousMillis = 0;

void setup()
{
    //loop for LED segments
    for (int i = 0; i < 7; i++)
    {
        //configure LEDs as OUTPUTS
        pinMode(LED[i], OUTPUT);
    }
    //loop for digits
    for (int j = 0; j < 4; j++)
    {
        //configure as outputs
        pinMode (DIGITS[j], OUTPUT);

        //all digits are turned off at first
        digitalWrite (DIGITS[j], DOWN);
    }
}

void Display(int pattern)
{
    //set the appropriate segments ON/OFF based on the bit pattern
    for (int i = 0; i < 7; i++)
    {
        //checks if the segment should be turned ON or OFF
        digitalWrite(LED[i], (pattern & (1 << i)) ? ON : OFF);
    }
};

//function to display entire uptime

```

```

void displayNumber(int minutes, int seconds)
{
    //extract digits from minutes and seconds
    int Dig1 = minutes / 10;    //tens place of minutes
    int Dig2 = minutes % 10;    //ones place of minutes
    int Dig3 = seconds / 10;    //tens place of seconds
    int Dig4 = seconds % 10;    //ones place of seconds

    //display the first digit (Dig1)
    Display(segmentStates[Dig1]);    //get the segment pattern
    digitalWrite(DIGITS[0], UP);    //enable DIG1
    digitalWrite(DIGITS[0], DOWN);    //disable DIG1

    //display the second digit (Dig2)
    Display(segmentStates[Dig2]);    //get the segment pattern
    digitalWrite(DIGITS[1], UP);    //enable DIG2

    //Decimal Point ON
    digitalWrite(13, ON);

    digitalWrite(DIGITS[1], DOWN);    //disable DIG2

    //Decimal Point OFF
    digitalWrite(13, OFF);

    //display the third digit (Dig3)
    Display(segmentStates[Dig3]);    //get the segment pattern
    digitalWrite(DIGITS[2], UP);    //enable DIG3
    //delay(3);    //short delay
    digitalWrite(DIGITS[2], DOWN);    //disable DIG3

    //display the fourth digit (Dig4)
    Display(segmentStates[Dig4]);    //get the segment pattern
    digitalWrite(DIGITS[3], UP);    //enable DIG4
    digitalWrite(DIGITS[3], DOWN);    //disable DIG4
};

void loop()
{
    //call a function that gives the current time in milliseconds
    //since the program started
    unsigned long currentMillis = millis();
    //check if a second passed
    //if the interval *1000* is changed, the counter goes slower or faster
    if (currentMillis - previousMillis >= 1000)
    {
        //update to current time

```

```

previousMillis = currentMillis;

//increment seconds and handle overflow
seconds++;
if (seconds > 59) //reset seconds after reaching 59
{
    seconds = 0;
    minutes++;
    if (minutes > 99) //reset minutes after reaching 99
    {
        minutes = 0;
    }
}

}
// display the current minutes and seconds
displayNumber(minutes, seconds);
}

```

Logbook from date to date

Day	Logbook of work
Sunday, 06.10.2024	First, I did a counter from 0 to 9 using only a digit.
Monday, 07.10.2024	I made the code for the 7-segment display with 4 digits.
Tuesday, 08.10.2024	I did the hardware part on the breadboard.
Wednesday, 09.10.2024 Thursday, 10.10.2024	I changed the wires and remade the hardware part, but it still didn't work.
Sunday, 13.10.2024	I tried again to fix the project.
Monday, 14.10.2024	I finally managed to fix the issue with the hardware part and the code.

Week 4 – Self reflection

This week, while doing the Computer Science assignments, I learned how to put different filters on pictures. Besides, I learned how to recover a picture from a memory card. I didn't find it difficult, because CS50 gave me enough information to manage to do the projects.

On the other hand, the assignment for Embedded Systems class gave me a hard time. By using a 7-segment display with 4 digits, we had to make an uptime counter. The first two digits from left to right represented minutes, while the last two digits represented seconds. It didn't seem to be a hard assignment to bring to life, but I had a problem with placing the resistors on the breadboard. In the end, I managed to fix the issues and build the project correctly.

This picture shows how an uptime counter on a 7-segment display with only one digit should work:

Week 5 – Data Structures

Inheritance

Complete the implementation of `inheritance.c`, such that it creates a family of a specified generation size and assigns blood type alleles to each family member. The oldest generation will have alleles assigned randomly to them.

The `create_family` function takes an integer (generations) as input and should allocate (as via `malloc`) one person for each member of the family of that number of generations, returning a pointer to the person in the youngest generation.

For example, `create_family(3)` should return a pointer to a person with two parents, where each parent also has two parents.

Each person should have alleles assigned to them. The oldest generation should have alleles randomly chosen (as by calling the `random_allele` function), and younger generations should inherit one allele (chosen at random) from each parent.

Each person should have parents assigned to them. The oldest generation should have both parents set to `NULL`, and younger generations should have parents be an array of two pointers, each pointing to a different parent.

This is the code for the assignment:

```
// Simulate genetic inheritance of blood type

#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

```

// Each person has two parents and two alleles
typedef struct person
{
    struct person *parents[2];
    char alleles[2];
} person;

// simulating 3 generations
const int GENERATIONS = 3;
// generating from grandchildren to grandparents
const int INDENT_LENGTH = 4;

person *create_family(int generations);
void print_family(person *p, int generation);
void free_family(person *p);
char random_allele();

int main(void)
{
    // Seed random number generator
    srand(time(0));

    // Create a new family with three generations
    person *p = create_family(GENERATIONS);

    // Print family tree of blood types
    print_family(p, 0);

    // Free memory
    free_family(p);
}

// Create a new individual with `generations`
person *create_family(int generations)
{
    // Allocate memory for new person
    person *new_person = malloc(sizeof(person));

    // If there are still generations left to create
    if (generations > 1)
    {
        // Create two new parents for current person by recursively calling create_family
        person *parent0 = create_family(generations - 1);
        person *parent1 = create_family(generations - 1);

        // Set parent pointers for current person
        new_person->parents[0] = parent0;
    }
}

```

```

        new_person->parents[1] = parent1;

        // Randomly assign current person's alleles based on the alleles of their parents
        new_person->alleles[0] = parent0->alleles[rand() % 2];
        new_person->alleles[1] = parent1->alleles[rand() % 2];
    }

    // If there are no generations left to create
    else
    {
        // Set parent pointers to NULL
        new_person->parents[0] = NULL;
        new_person->parents[1] = NULL;

        // TODO: Randomly assign alleles
        new_person->alleles[0] = random_allele();
        new_person->alleles[1] = random_allele();
    }

    // Return newly created person
    return new_person;
}

// Free `p` and all ancestors of `p`.
void free_family(person *p)
{
    // Handle base case
    if (p == NULL)
    {
        return;
    }

    // Free parents recursively
    free_family(p->parents[0]);
    free_family(p->parents[1]);

    // Free child
    free(p);
}

// Print each family member and their alleles.
void print_family(person *p, int generation)
{
    // Handle base case
    if (p == NULL)
    {
        return;
    }

```

```

}

// Print indentation
for (int i = 0; i < generation * INDENT_LENGTH; i++)
{
    printf(" ");
}

// Print person
if (generation == 0)
{
    printf("Child (Generation %i): blood type %c%c\n", generation, p->alleles[0],
        p->alleles[1]);
}
else if (generation == 1)
{
    printf("Parent (Generation %i): blood type %c%c\n", generation, p->alleles[0],
        p->alleles[1]);
}
else
{
    for (int i = 0; i < generation - 2; i++)
    {
        printf("Great-");
    }
    printf("Grandparent (Generation %i): blood type %c%c\n", generation, p->alleles[0],
        p->alleles[1]);
}

// Print parents of current generation
print_family(p->parents[0], generation + 1);
print_family(p->parents[1], generation + 1);
}

// Randomly chooses a blood type allele.
char random_allele()
{
    int r = rand() % 3;
    if (r == 0)
    {
        return 'A';
    }
    else if (r == 1)
    {
        return 'B';
    }
    else

```



```
{  
    return '0';  
}
```

Logbook from date to date

Day	Logbook of work
Wednesday, 09.10.2024	Done the assignment.

The Harvard score with the title of the program and my name:

[me50](#) / [users](#) / [Alexia220700](#) / [cs50](#) / [problems](#) / [2024](#) / [x](#) / [inheritance](#)

🔑 #1 submitted 3 minutes ago, Wednesday, October 9, 2024 3:17 PM CEST
check50 7/7 • style50 1.00 • 0 comments
[tar.gz](#) • [zip](#)

<https://submit.cs50.io/check50/750a563b9fefae2cb8e28d42bd1efabae2433453>

Week 5 – Jumbo Clock

The LCD display should be set up to show the time in the format of hh:mm, where the time will appear as two digits for the hour and two digits for the minute, separated by a colon.

When the reset button is pressed, the system generates a random time. This time will consist of a random hour between 00 and 23 and a random minute between 00 and 59.

This is the code for the assignment:

```
#include <LCD_I2C.h>
LCD_I2C lcd(0x27, 16, 2);

//timer variables
int seconds = 0; //used to properly increment minutes
int minutes;
int hours;
unsigned long previousMillis = 0;

//custom characters for parts of the digits
byte charZero[8] =
{
    B11111,
    B10000,
    B10000,
    B10000,
    B10000,
    B10000,
    B10000,
    B10000,
    B11111
};

byte charOne[8] =
{
    B11111,
    B00001,
    B00001,
    B00001,
    B00001,
    B00001,
    B00001,
    B00001,
}
```

```

    B11111
};

byte charTwo[8] =
{
    B10000,
    B10000,
    B10000,
    B10000,
    B10000,
    B10000,
    B10000,
    B10000,
    B11111
};

byte charThree[8] =
{
    B00001,
    B00001,
    B00001,
    B00001,
    B00001,
    B00001,
    B00001,
    B00001,
    B11111
};

byte charFour[8] =
{
    B11111,
    B10000,
    B10000,
    B10000,
    B10000,
    B10000,
    B10000,
    B10000,
    B10000
};

byte charFive[8] =
{
    B11111,
    B00001,
    B00001,
    B00001,
    B00001,
    B00001,
    B00001,
    B00001,

```

```

    B00001
};

byte charSix[8]
{
    B00001,
    B00001,
    B00001,
    B00001,
    B00001,
    B00001,
    B00001,
    B00001,
    B00001
};

byte charTwoPoints[8] =
{
    B00000,
    B00000,
    B00000,
    B00000,
    B00000,
    B00000,
    B00000,
    B00000,
    B00000,
};

//assembling the pieces as numbers
const byte digits[10][2][2] =
{
    //number zero
    {
        {4, 5}, {2, 3}
    },
    //number one
    {
        {7, 6}, {7, 6}
    },
    //number two
    {
        {7, 1}, {7, 2}
    },
    //number three
    {
        {7, 1}, {7, 3}
    },

    //number four

```

```

{
    {2, 3}, {7, 6}
},
//number five
{
    {0, 7}, {3, 7},
},
//number six
{
    {4, 7}, {0, 1}
},
//number seven
{
    {4, 5}, {7, 6}
},
//number eight
{
    {0, 1}, {0, 1}
},
//number nine
{
    {0, 1}, {7, 3}
}
};

void setup()
{
    //put your code here to run once
    lcd.begin();
    lcd.backlight();
    lcd.clear();

    //set a random time for the clock
    hours = random (0, 24);
    minutes = random (0, 59);

    //load custom characters into the LCD
    lcd.createChar(0, charZero);
    lcd.createChar(1, charOne);
    lcd.createChar(2, charTwo);
    lcd.createChar(3, charThree);
    lcd.createChar(4, charFour);
    lcd.createChar(5, charFive);
    lcd.createChar(6, charSix);
    lcd.createChar(7, charTwoPoints);

    updateDisplay(hours, minutes);
}

```

```

void loop()
{
    //call a function that gives the current time in milliseconds
    //since the program started
    unsigned long currentMillis = millis();

    //check if a second passed
    //if the interval *1000* is changed, the counter goes slower or faster
    if (currentMillis - previousMillis >= 10)
    {
        //update to current time
        previousMillis = currentMillis;

        //increment seconds and handle overflow
        seconds++;
        if (seconds > 59) //reset seconds after reaching 59
        {
            seconds = 0;
            minutes++;
            if (minutes > 59) //reset minutes after reaching 99
            {
                minutes = 0;
                hours++;

                if(hours > 23)
                {
                    hours = 0;
                }
            }
        }
        updateDisplay(hours, minutes);
    }
}

//update the digits of the Jumbo Clock
void updateDisplay(int hours, int minutes)
{
    int DIG1 = hours / 10;
    int DIG2 = hours % 10;
    int DIG3 = minutes / 10;
    int DIG4 = minutes % 10;

    //display first two digits, for hours
    displayDigit(0 , 0, DIG1);
    displayDigit(3, 0, DIG2);
}

```

```

//add the : char
lcd.setCursor(6, 0);
lcd.write('.');
lcd.setCursor(6, 1);
lcd.write('.');

//display the next two digits, for minutes
displayDigit(8, 0, DIG3);
displayDigit(11, 0, DIG4);
}

//function to display a digit at (x, y) points on the LCD
void displayDigit(int x, int y, int digit)
{
    //display segment top left
    lcd.setCursor(x, y);
    lcd.write(digits[digit][0][0]);

    //display segment top right
    lcd.setCursor(x + 1, y);
    lcd.write(digits[digit][0][1]);

    //display segment bottom left
    lcd.setCursor(x, y + 1);
    lcd.write(digits[digit][1][0]);

    //display segment bottom right
    lcd.setCursor(x + 1, y + 1);
    lcd.write(digits[digit][1][1]);
}

```

Week 5 – Self reflection

Inheritance was the assignment we had to complete for our Computer Science class. It is a program that simulates the inheritance of blood types for each member of a family. In this program, a person's blood type is determined by two alleles, which can be A, B, or O. Each child randomly inherits one allele from each parent. The program should determine the possible blood type combinations for every individual in the family, considering how alleles are passed down from parents to children. Writing this program helped us understand genetic inheritance in a practical way.

For Embedded Systems class, we had to build a Jumbo Clock. This assignment required us to display custom numbers designed by us on the screen. Every time the Arduino was plugged into the laptop, or the reset button was pressed, it had to show a random time. The hour was required to be between 00 and 23, and the minutes between 00 and 59. In my opinion, this assignment wasn't too difficult to create, because it was mostly the same concept as the Uptime counter, which we had already worked on.

Week 6 – Speller

For this program, we had to implement a program that spell-checks a file by loading a dictionary of words from disk into memory, using a hash table.

This is the code for the assignment:

```
#include <ctype.h> //for character handling functions
#include <math.h> //for mathematical functions
#include <stdbool.h> //for boolean
#include <stdint.h> //for fixed-width integers type
#include <stdio.h> //for input/output functions
#include <stdlib.h> //for memory allocation
#include <string.h> //for string manipulation
#include <strings.h> //for string comparison

#include "dictionary.h"

//define the number of buckets in the hash table
#define N 20000

// Represents a node in a hash table
typedef struct node
{
    char word[LENGTH + 1];
    struct node *next;
} node;

// declares hash table
node *table[N];

//initialize word_count for dictionary
unsigned int word_count = 0;

// Hash function prototype
unsigned int hash(const char *word);

// Function to load the dictionary
bool load(const char *dictionary)
{
    // Open dictionary file
    FILE *file = fopen(dictionary, "r");
    if (file == NULL)
    {
```

```

        return false;
    }

    //buffer for a word
    //the additional character is for the null terminator "\0"
    char word[LENGTH + 1];

    //reads words from the dictionary file
    //until EOF, end of file
    while (fscanf(file, "%s", word) != EOF)
    {
        //creates a new node for each word
        //and allocates memory for it
        node *new_node = malloc(sizeof(node));

        //if the new node is empty
        if (new_node == NULL)
        {
            return false;
        }

        //copy the word into the new node
        strcpy(new_node->word, word);

        //hash the word to obtain a hash value
        unsigned int index = hash(word);

        //insert node into the hash table
        //sets next pointer to point to the current head of the linked list
        new_node->next = table[index];
        //updates the head of the linked list to point to the new node
        table[index] = new_node;

        //increment word count
        word_count++;
    }

    //close dictionary file
    fclose(file);
    return true;
}

// Function to check if a word is in the dictionary
bool check(const char *word)
{
    //hash the word to obtain a hash value
    unsigned int index = hash(word);

```

```

//access linked list at that index in the hash table
node *cursor = table[index];

//go through the linked list, looking for the word
while (cursor != NULL)
{
    //compare word in dictionary to input word
    //case-insensitive comparison
    if (strcasecmp(word, cursor->word) == 0)
    {
        //word found
        return true;
    }
    //updates cursor to point to the next node in the linked list
    cursor = cursor->next;
}

//word not found
return false;
}

// Hashes word to a number
unsigned int hash(const char *word)
{
    //hash the first letter of the word
    //convert the first letter of the word to lowercase
    return tolower(word[0]) - 'a';
}

// Function to return the number of words in the dictionary
unsigned int size(void)
{
    return word_count;
}

// Function to unload the dictionary from memory
bool unload(void)
{
    //iterate over the hash table
    for (int i = 0; i < N; i++)
    {
        // Free linked lists
        node *cursor = table[i];
        while (cursor != NULL)
        {
            //store current node

```

```

node *temp = cursor;
//move to the next node
cursor = cursor->next;
//free the memory of the current node
free(temp);
}
}
return true;
}

```

Logbook from date to date

Day	Logbook of work
Wednesday, 16.10.2024	I started with the bool load void.
Thursday, 17.10.2024	I kept working on bool load and size void.
Friday, 18.10.2024	I did the hash function.
Monday, 21.10.2024	I did the bool unload.
Tuesday, 22.10.2024	Done the assignment.

The Harvard score with the title of the program and my name:

[me50](#) / [users](#) / [Alexia220700](#) / [cs50](#) / [problems](#) / [2024](#) / [x](#) / [speller](#)

🔑 #1 submitted 5 hours ago, Tuesday, October 22, 2024 3:25 PM CEST
 check50 9/9 • style50 1.00 • 0 comments
[tar.gz](#) • [zip](#)

<https://submit.cs50.io/check50/07eba859d4633525a109eb11364850a8f422a75c>

Week 6 – Weather Station

For this assignment, we had to create a Weather Station using the Arduino Uno R4 Wi-Fi. This Weather Station had to display the minimum temperature, the maximum temperature, the current temperature/humidity. In addition, it had to display a moon if it is night, or a sun if it is day, using a light sensor.

This is the code for the assignment:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "DHT.h"

// LCD setup (16x2 LCD at I2C address 0x27)
LiquidCrystal_I2C lcd(0x27, 16, 2);

// button pin
Int Button = 3;
int pressed = 0;
unsigned long pressStartTime;
bool longPressDetected = false;

// store max and min values
int minIn = 100;
int minOut = 100;
int minHum = 100;
int maxIn = -100;
int maxOut = -100;
int maxHum = -100;

// Pin setup for DHT sensor
#define DHTPIN 2 // DHT sensor connected to pin 2
#define DHTTYPE DHT22 // Using the DHT11 sensor

DHT dht(DHTPIN, DHTTYPE);

//moon char
byte moonChar[8] =
```

```

{
  B11100,
  B01110,
  B00111,
  B00111,
  B00111,
  B00111,
  B01110,
  B11100
};

//sun char
byte sunChar[8] =

{
  B00000,
  B00100,
  B01110,
  B11111,
  B01110,
  B00100,
  B00000,
  B00000
};

// the setup routine runs once when you press reset:
void setup() {

  // Initialize serial communication at 9600 bits per second
  Serial.begin(9600);

  // Initialize LCD
  lcd.init();
  lcd.backlight();

  lcd.createChar(0, moonChar);      //create character with index 0
  lcd.createChar(1, sunChar);      //create character with index 1

  // Initialize DHT sensor
  dht.begin();

  // setup button
  pinMode(Button, INPUT_PULLUP);
  //interrupt for a button press
  attachInterrupt(digitalPinToInterrupt(Button), press, FALLING);
}

```

```

// push button function
void press()
{
    //calculate how long a button has been pressed
    unsigned long pressDuration = millis() - pressStartTime;

    if (pressDuration > 2000)
    {
        longPressDetected = true; // Detect long press if held for more than 2 seconds
    }
    else
    {
        longPressDetected = false;

        // track the current state of a button press
        // the button is not currently pressed
        if (pressed == 0)
        {
            pressed = 1;
        }
        // the button has been pressed once
        else if (pressed == 1)
        {
            pressed = 2;
        }
        // the button has been pressed and released
        else if (pressed == 2)
        {
            pressed = 0;
        }
        // store the current time (in milliseconds)
        pressStartTime = millis();
    }
}

// the loop routine runs repeatedly:
void loop() {

    // Read temperature from LM35 sensor on analog pin A0
    int lm35Value = analogRead(A0);

    // Convert voltage to temperature for LM35
    float lm35Voltage = lm35Value * (5.0 / 1023.0);
    int lm35Temp = lm35Voltage * 100;

    // set min and max values for out temp
    if (lm35Temp < minOut)

```

```

{
    minOut = lm35Temp;
}

if (lm35Temp > maxOut)
{
    maxOut = lm35Temp;
}

// Read temperature and humidity from DHT sensor
int humidity = dht.readHumidity();

int dhtTempC = dht.readTemperature();    // Read temperature as Celsius

// set min and max values for in temp and humidity
if (humidity < minHum)
{
    minHum = humidity;
}

if (humidity > maxHum)
{
    maxHum = humidity;
}

if (dhtTempC < minIn)
{
    minIn = dhtTempC;
}

if (dhtTempC > maxIn)
{
    maxIn = dhtTempC;
}

// Check if any reads failed from DHT
if (isnan(humidity) || isnan(dhtTempC)) {

    Serial.println("Failed to read from DHT sensor!");

    Return;
}

// Print readings to the Serial Monitor
Serial.print("LM35 Temperature: ");

```



```

Serial.println(lm35Temp);

Serial.print("Humidity (DHT): ");

Serial.println(humidity);

Serial.print("Temperature (DHT): ");

Serial.println(dhtTempC);


// reads the input on analog pin A0 (value between 0 and 1023)
if (!longPressDetected)
{
  if (pressed == 0)
  {
    lcd.setCursor(0, 0);

    lcd.print("Out:");

    lcd.print(lm35Temp);

    lcd.print((char)223); // Degree symbol

    lcd.print("C ");

    lcd.print("In:");

    lcd.print(dhtTempC);

    lcd.print((char)223); // Degree symbol

    lcd.print("C");

    // Display DHT humidity and heat index on the LCD
    lcd.setCursor(0, 1);

    lcd.print("Humidity: ");

    lcd.print(humidity);

    lcd.print("% ");

    // capture the current analog voltage on pin A1 and store it in analogValue for
    further processing

```

```

int analogValue = analogRead(A1);

Serial.print("Analog reading: ");

Serial.print(analogValue);    // the raw analog reading

lcd.setCursor(15, 1);          // column 15, row 1

// We'll have a few thresholds, qualitatively determined
if (analogValue < 100)

{

    lcd.write(0);

}

else if (analogValue >= 100)

{

    lcd.write(1);

}

}

// print max values on lcd
else if (pressed == 1)
{

    lcd.setCursor(0, 0);

    lcd.print("Out:");

    lcd.print(maxOut);

    lcd.print((char)223); // Degree symbol

    lcd.print("C ");

    lcd.print("In:");

    lcd.print(maxIn);

```

```

    lcd.print((char)223); // Degree symbol

    lcd.print("C");

    lcd.setCursor(0, 1);

    lcd.print("Humidity: ");

    lcd.print(maxHum);

    lcd.print("%max");
}

// print min values on lcd
else if (pressed == 2)
{
    lcd.setCursor(0, 0);

    lcd.print("Out:");

    lcd.print(minOut);

    lcd.print((char)223); // Degree symbol

    lcd.print("C ");

    lcd.print("In:");

    lcd.print(minIn);

    lcd.print((char)223); // Degree symbol

    lcd.print("C");

    lcd.setCursor(0, 1);

    lcd.print("Humidity: ");

    lcd.print(minHum);

    lcd.print("%min");
}
}

```

```
if (longPressDetected)
{
    minIn = 100;

    minOut = 100;

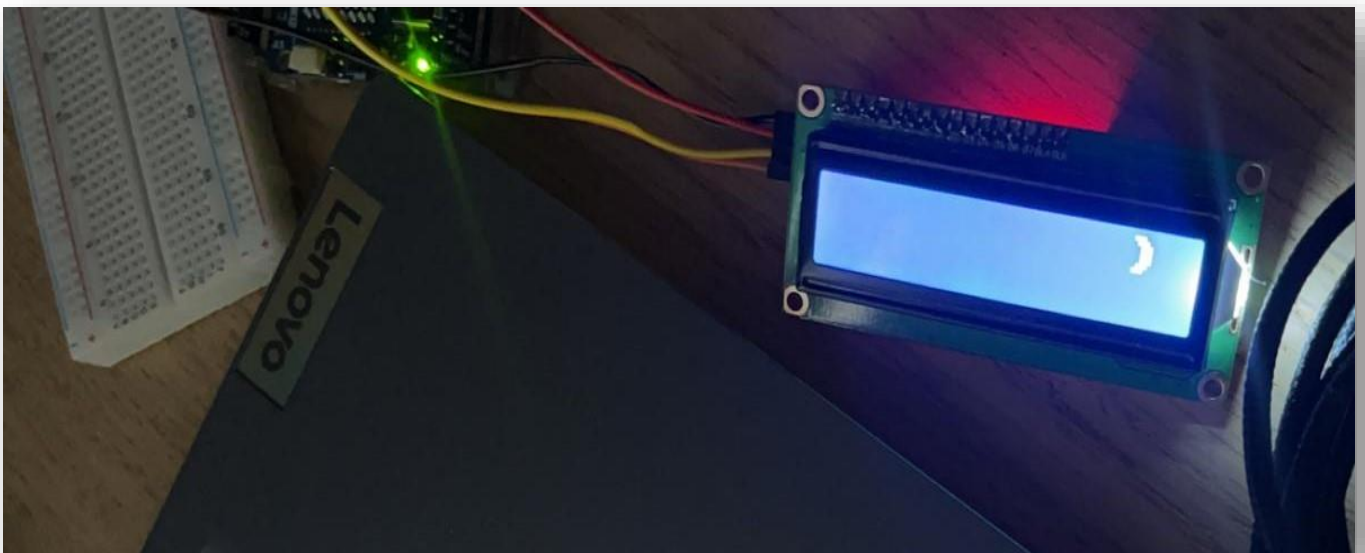
    minHum = 100;

    maxIn = -100;

    maxOut = -100;

    maxHum = -100;
}
delay(500);
}
```

This picture shows how the LDR understood that it is night, then displayed the specific custom character on the LCD:



Week 6 – Self reflection

For the Embedded Systems class we had to create a Weather Station. On the first screen, we had to display the current temperature and humidity, on the second screen we had to display the minimum temperature and humidity, and on the third screen we had to display the maximum temperature. This gave a full overview of the current and historical data of the weather conditions.

We also had to incorporate a light sensor, called LDR, to check what part of the day it is. We also had to incorporate a light sensor, called LDR, to detect the time of day. I had to create two custom characters: a moon and a sun. The moon was displayed when it was dark, indicating night-time, while the sun appeared during daylight. Designing these custom characters added an interesting visual component to the project, making it both functional and interactive.

For the Computer Science class, I found the assignment quite challenging. It took me a lot of time to complete. We had to implement a program that spell-checks a file by loading a dictionary of words from disk into memory using a hash table. The most difficult part was ensuring that the hash table functioned efficiently and handled collisions properly.

GitHub project – using PlatformIO

This program first blinks with the integrated LED on the Arduino UNO R4 Wi-Fi board. It then presents a Menu with an option between 1, which allows the user to add 2 numbers and receive their sum; and option 2, which allows the user to multiply 2 numbers and receive that value. After detecting Input from either function, the user will be re-prompted with the Menu function, and able to make another choice.

This is the code for the assignment:

```
#include <Arduino.h>

// test
// put function declarations here:
int myFunction(int, int);
int ledpin = 13;

void setup() {
// put your setup code here, to run once: please heckin work
int result = myFunction(2, 3);
}

void showMenu();
void Sum();
void Multiplication();
void Blink();

void loop() {
// put your main code here, to run repeatedly:
}

void setup()
{
// Start
Serial.begin(9600);

//put function definitions here:
int Sum() {
Serial.println("Provide the first number: ");
int x = Serial.parseInt();
```

```

// blink pinmode
pinMode(ledpin, OUTPUT);

// Show the menu
showMenu();
}

Serial.println("Provide the second number: ");
int y = Serial.parseInt();

int sum = y + x;
Serial.print("Sum is: ");
Serial.println(sum);

showmenu();
}
]
void loop()
{
Blink();

if (Serial.available() > 0)
{
// Read the user input
char option = Serial.read();

// Check option
switch (option)
{
case '1':
Sum();
break;
case '2':
Multiplication();
break;
default:
Serial.println("Invalid option. 1 or 2.");
showMenu(); // Show the menu again if its invalid
break;
}
}
}

// display the menu
void showMenu()
{

```

```

Serial.println("1: Calculate Sum");
Serial.println("2: Calculate Multiplication");
Serial.println("1 or 2? ");
}

void Blink()
{
digitalWrite(ledpin, HIGH);
delay(1000);
digitalWrite(ledpin, LOW);
delay(1000);
}

//used to make a sum of two numbers
void Sum()
{
int x;
int y;

Serial.println("Provide the first number: ");
while (Serial.available() == 0)
{
// wait for input
}
x = Serial.parseInt();
Serial.println(x);
Serial.println("Provide the second number: ");

while (Serial.available() == 0)
{
// wait for input
}
y = Serial.parseInt();
Serial.println(y);

int sum = x + y;
Serial.print("The Result Is: ");
Serial.println(sum);

delay(1500);
showMenu();
}

// Multiplication Function
void Multiplication()
{
// get 2 integers from input, return the numbers

```



```
int num1;
int num2;

Serial.println("Please Provide the First Number: ");

while (Serial.available() == 0)
{
  // wait for input
}

num1 = Serial.parseInt();
Serial.println(num1);

Serial.println("Please Provide the Second Number: ");

while (Serial.available() == 0)
{
}

num2 = Serial.parseInt();
Serial.println(num2);

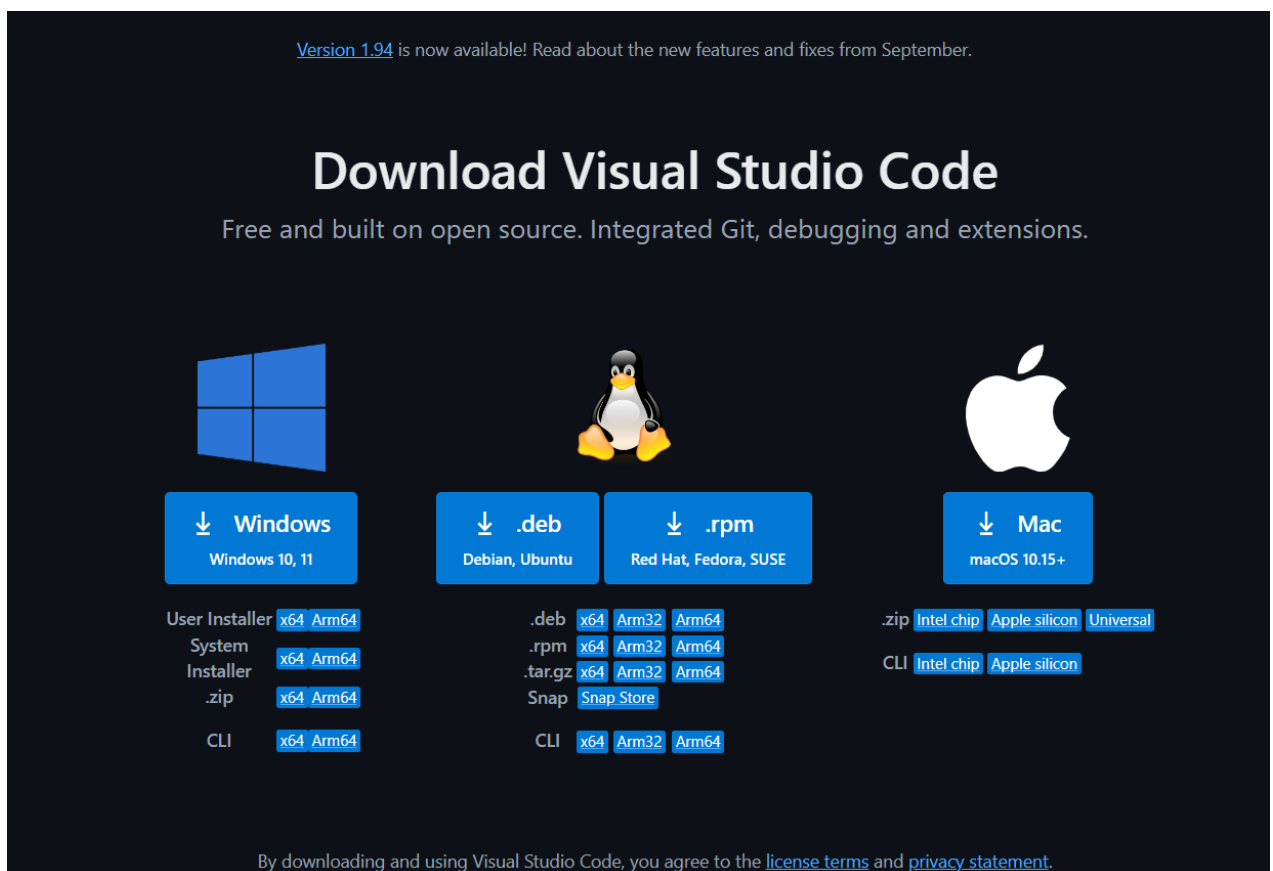
int Multiplied = num1 * num2;
Serial.print("The Result Is: ");
Serial.println(Multiplied);
delay(1500);

showMenu();
}
```

Set up guides

Visual Studio Code

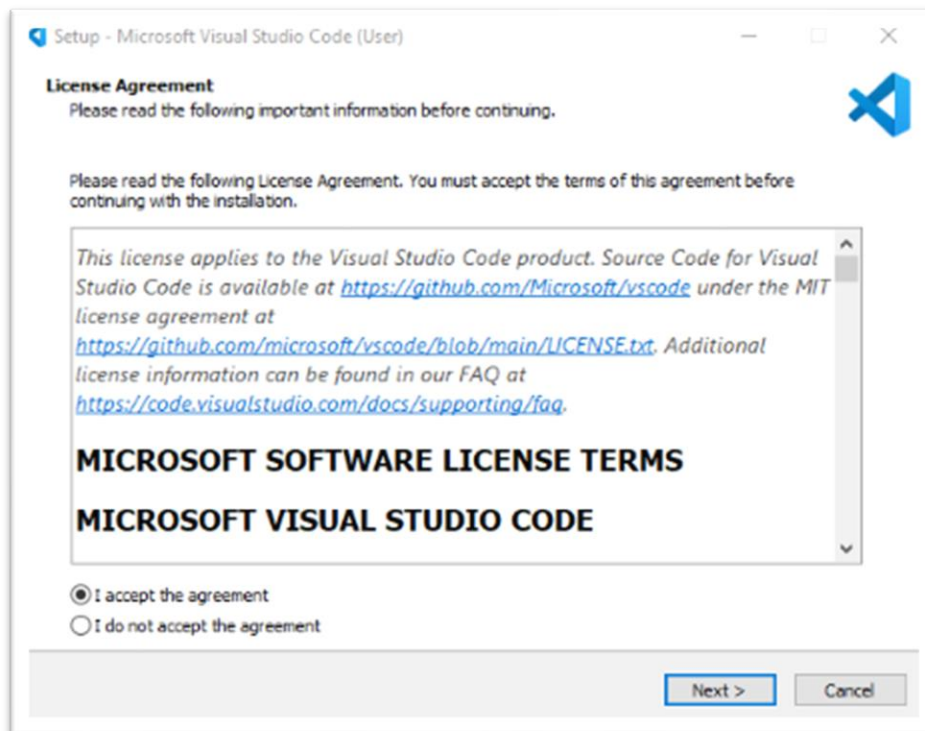
1. Go to [Download Visual Studio Code - Mac, Linux, Windows](#), the official website.
2. Choose one of the following versions:



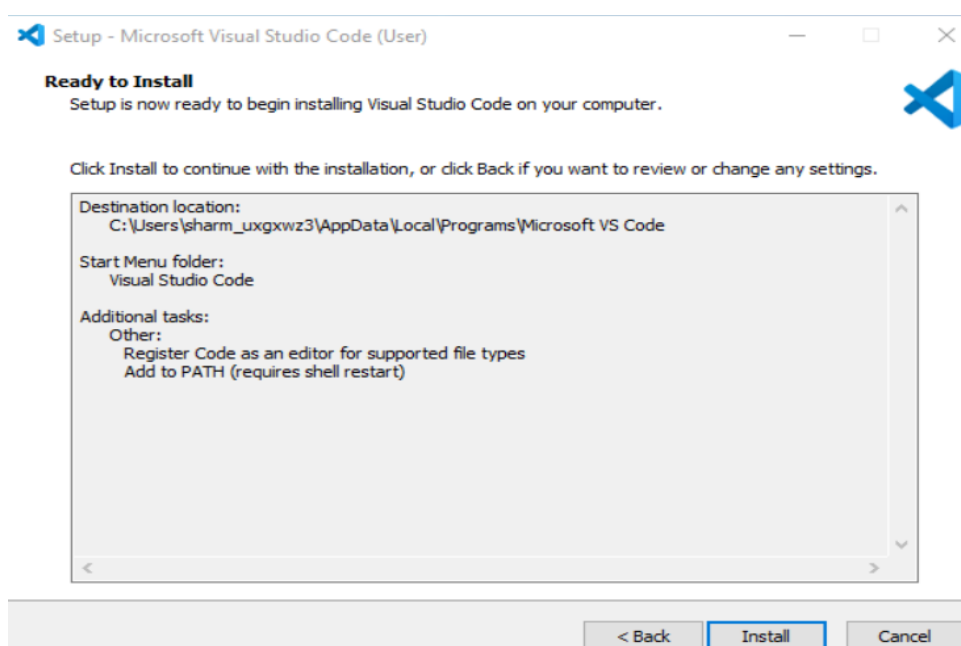
3. Execute the download file.



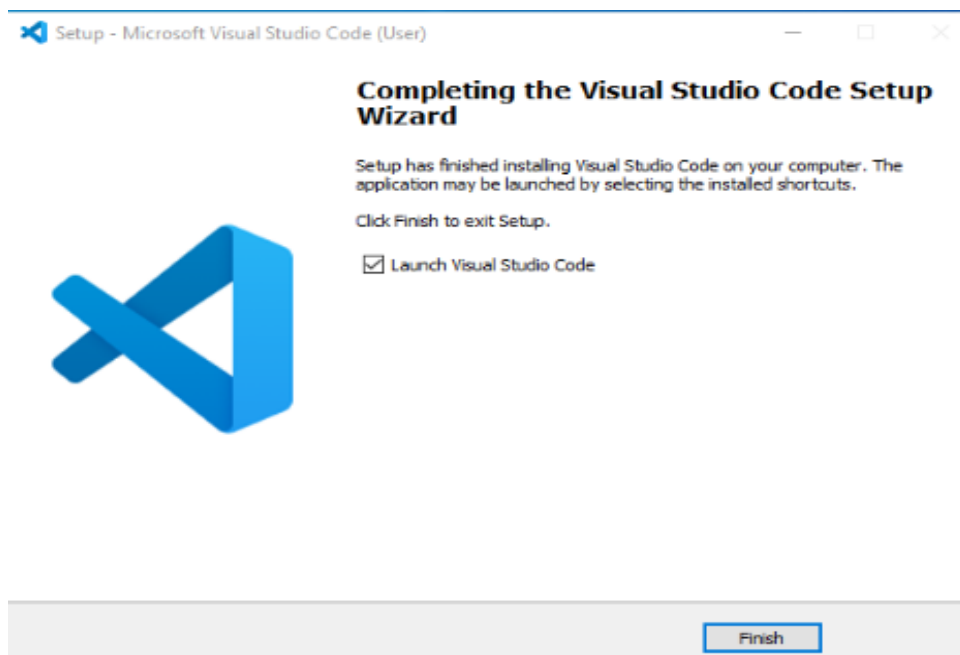
4. After the Installer opens, accept the Terms & Conditions, then click the next button.



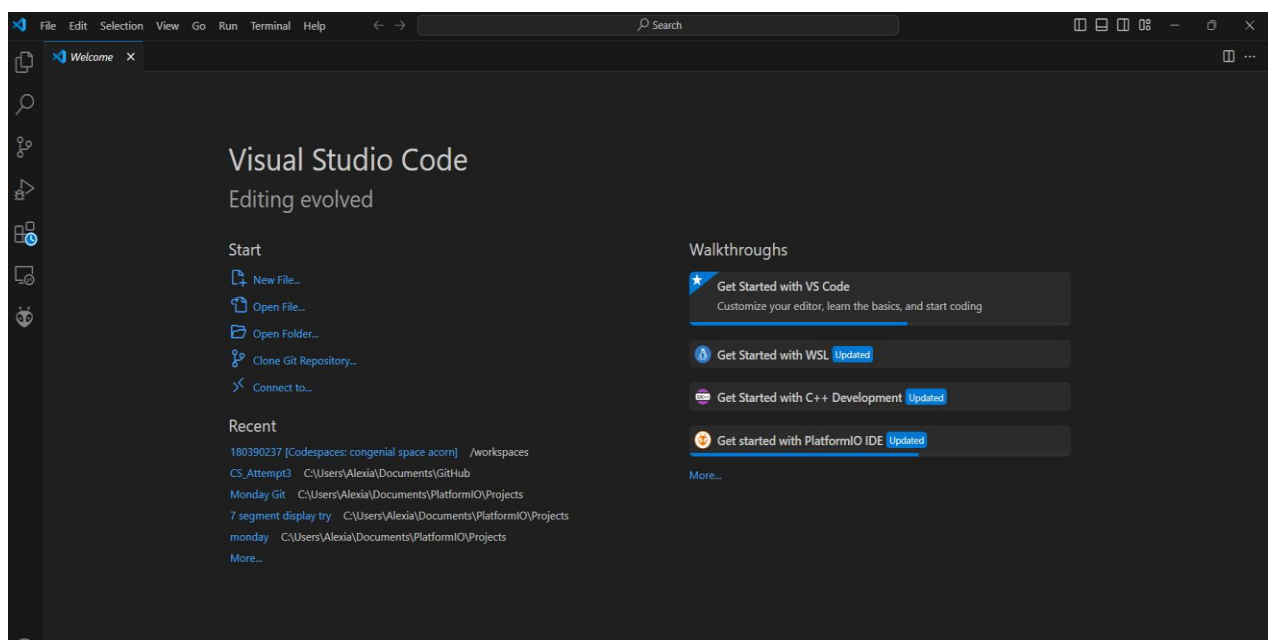
5. Then it will ask to begin the installation setup. Click on the Install button.



6. Click on the Launch button to start it.

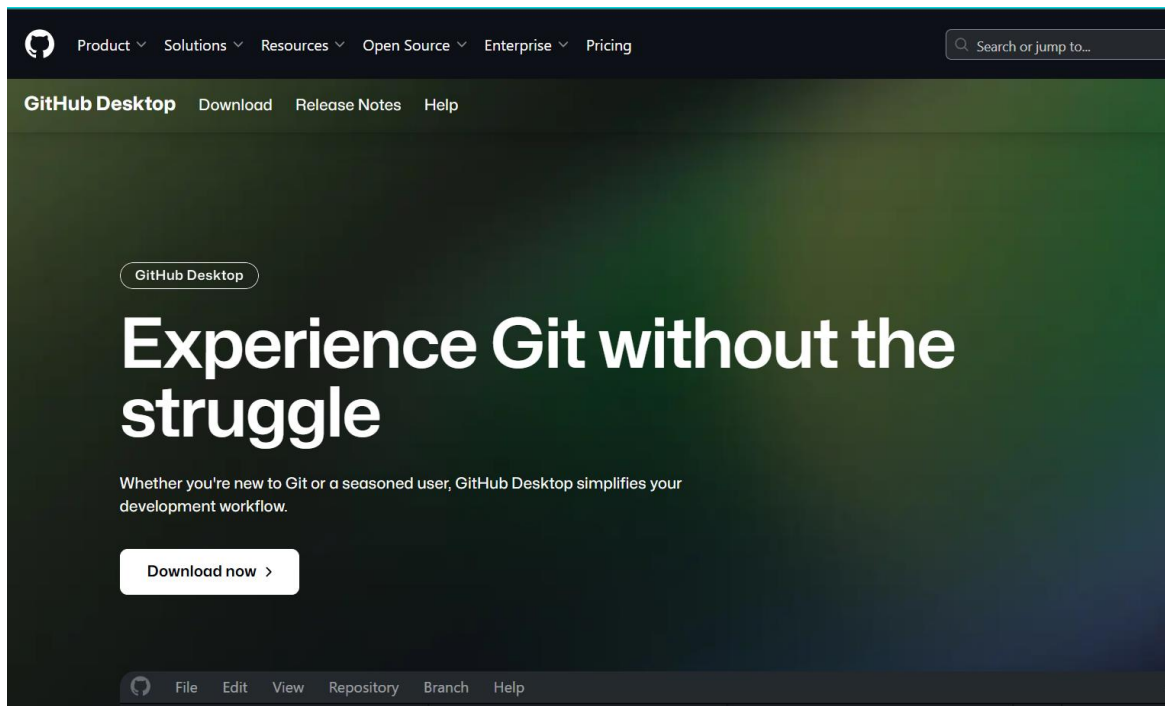


7. Now you can create a new file in the Visual Studio Code window:

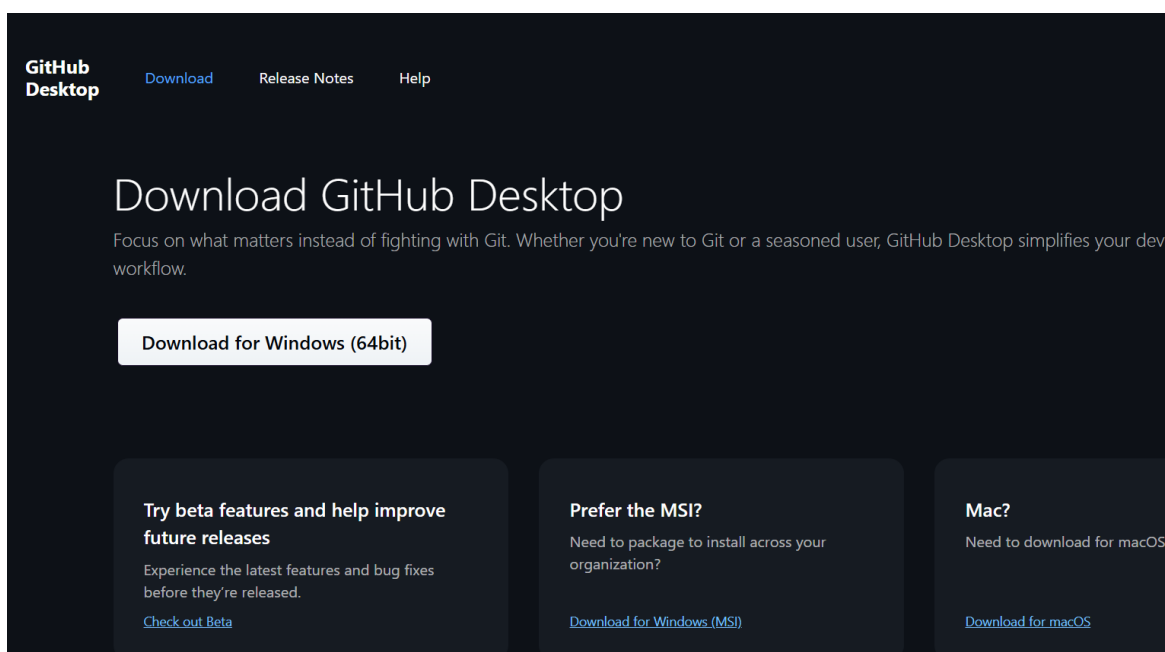


GitHub for Desktop

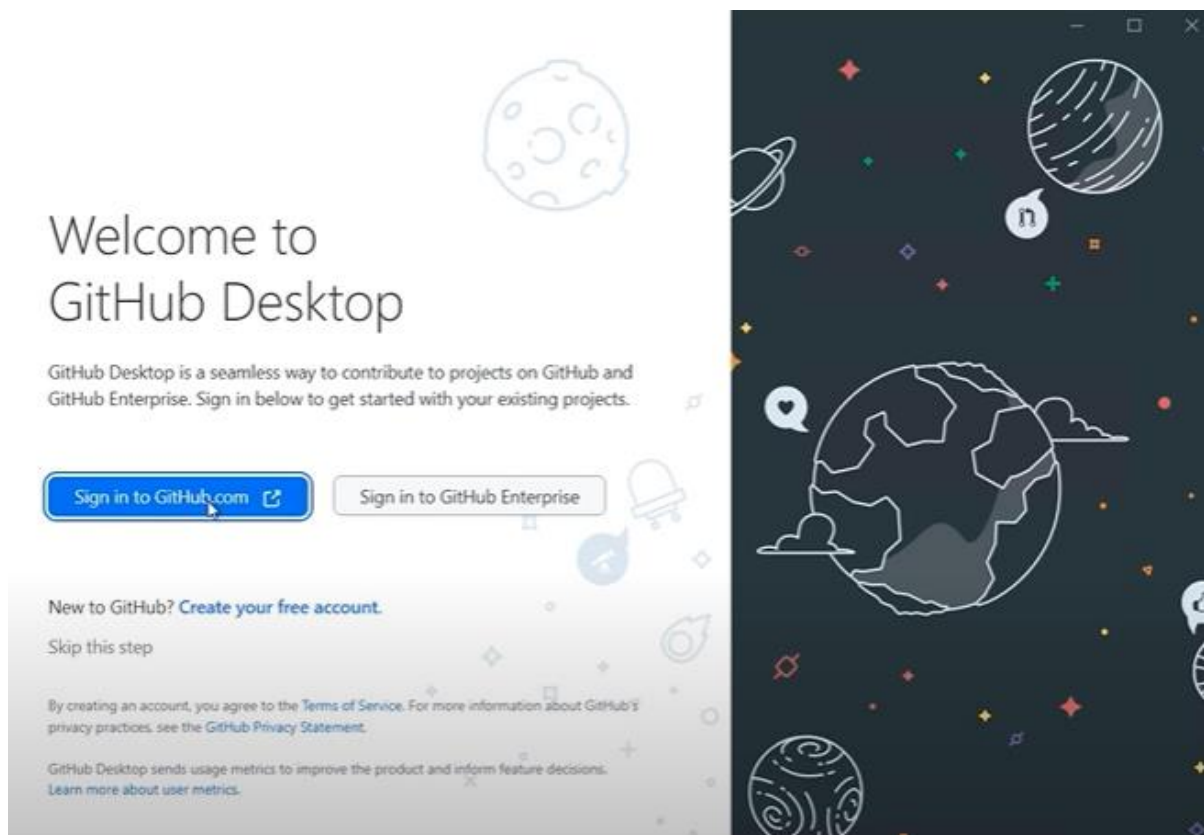
1. Go to the official GitHub website: <https://github.com/apps/desktop> and click on Download Now.



2. Choose the right version for you:



3. After installing GitHub Desktop, you will need to log into your GitHub account on the website.



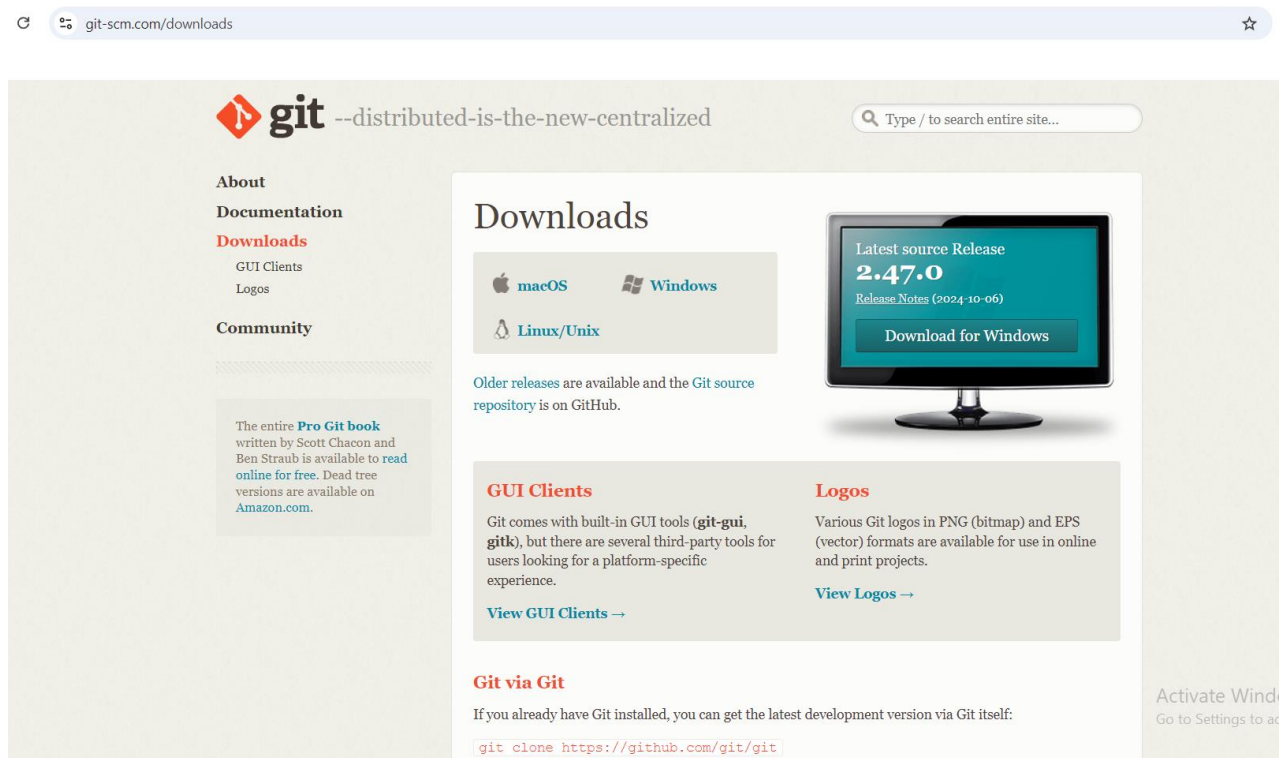
4. After installing Git too, you'll be able to use GitHub Desktop.

Git

- Git supports rapid branching and merging. It also includes specific tools for visualizing and navigating a non-linear development history in the most optimized way possible.
- It is efficient in handling large projects.

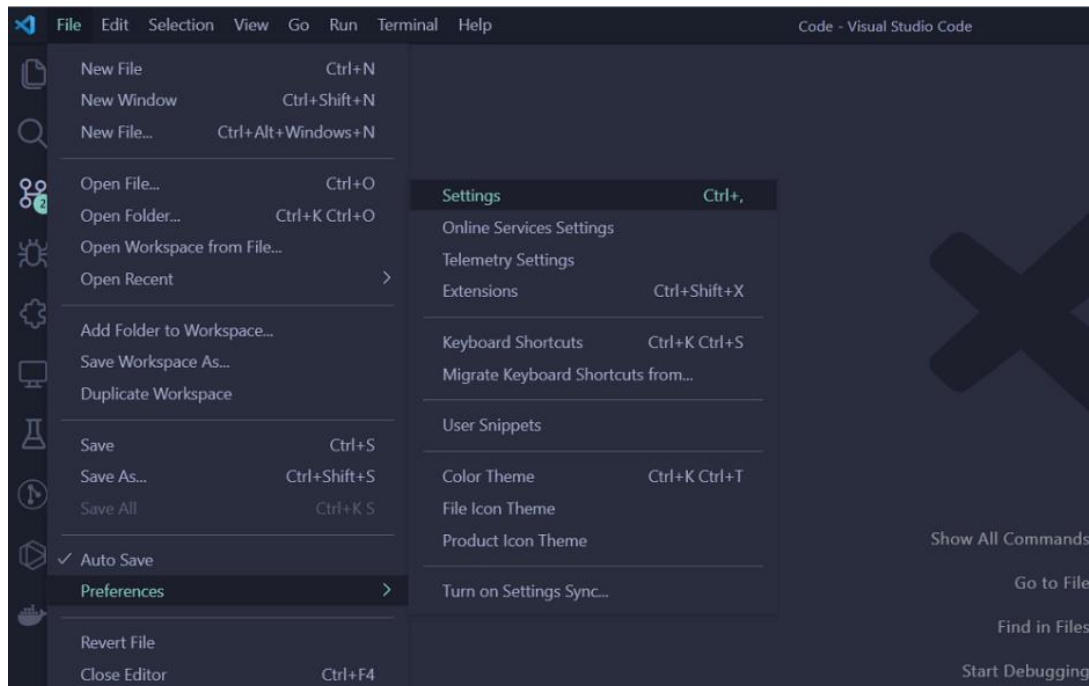
How to set up:

1. Go to Git official website: <https://git-scm.com/downloads> .

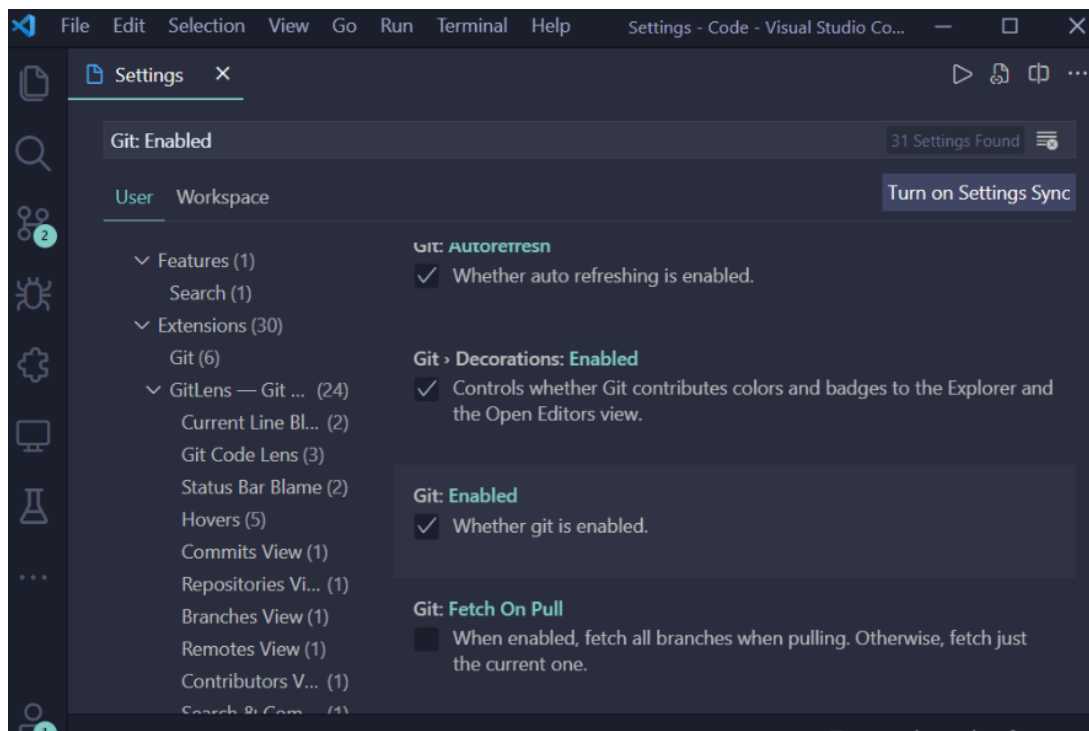


2. After Installing, you can check if it is installed properly or not by typing the following command in the Command Prompt: `git-version`.
3. Create an account on GitHub or log into your account.
4. Configure your Git account with your GitHub. To do this Open Command Prompt and Type:
 - `git config --global user.name "YourUserNameOnGithub"`
 - `git config --global user.email "YourEmail"`
 - ★ Replace **YourUserNameOnGithub** with your GitHub-username, and **YourEmail** with the email-id used while creating an account on GitHub.

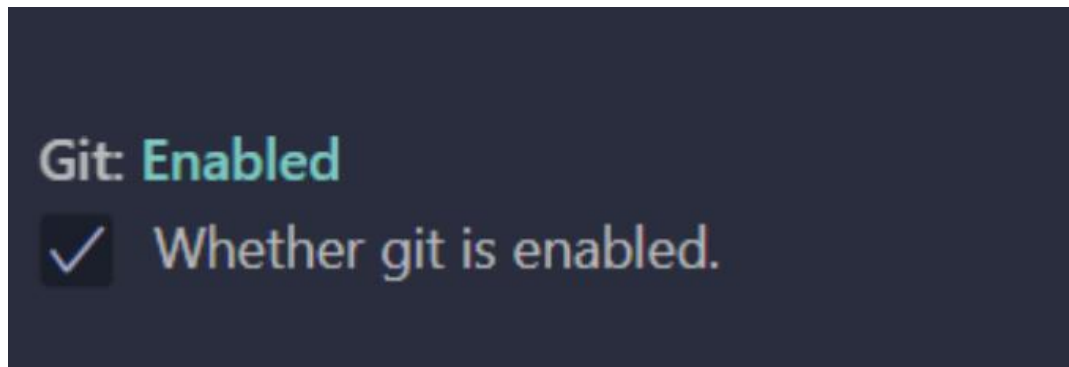
5. After configuring your details, your GitHub username and email will be displayed, this will make sure that your Git is linked with your GitHub. Type `git config --global --list` in the Command Prompt.
6. Open Visual Studio Code and go to settings.



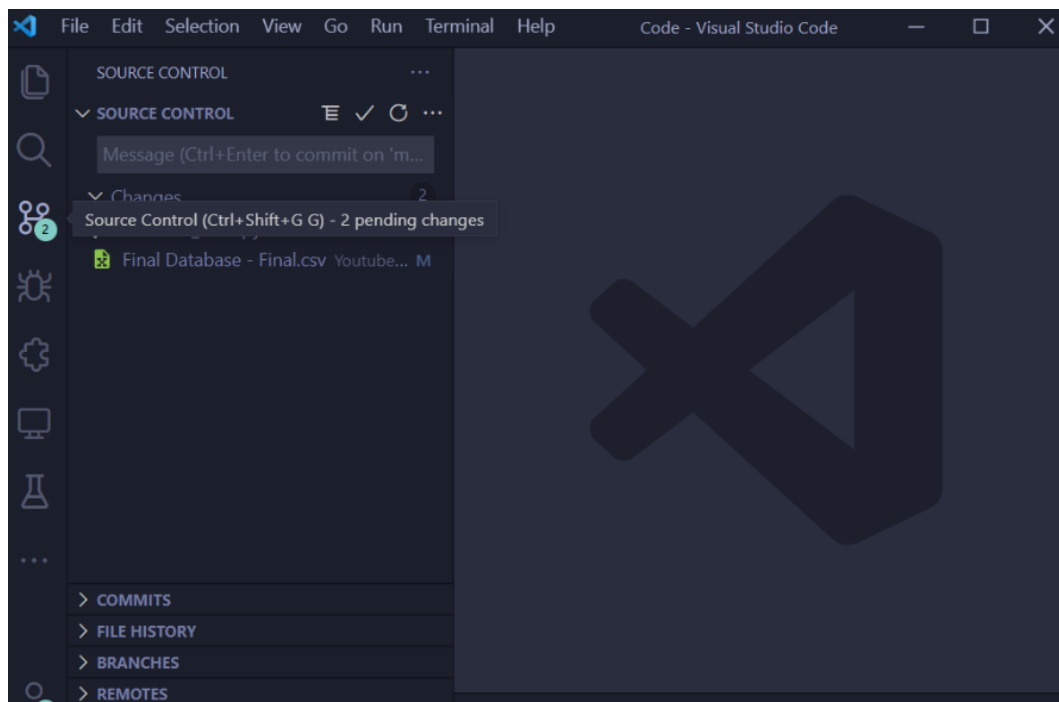
7. Now go to the Search Bar and type – *“Git: Enabled”*.



8. Tick the Checkbox to Enable Git.

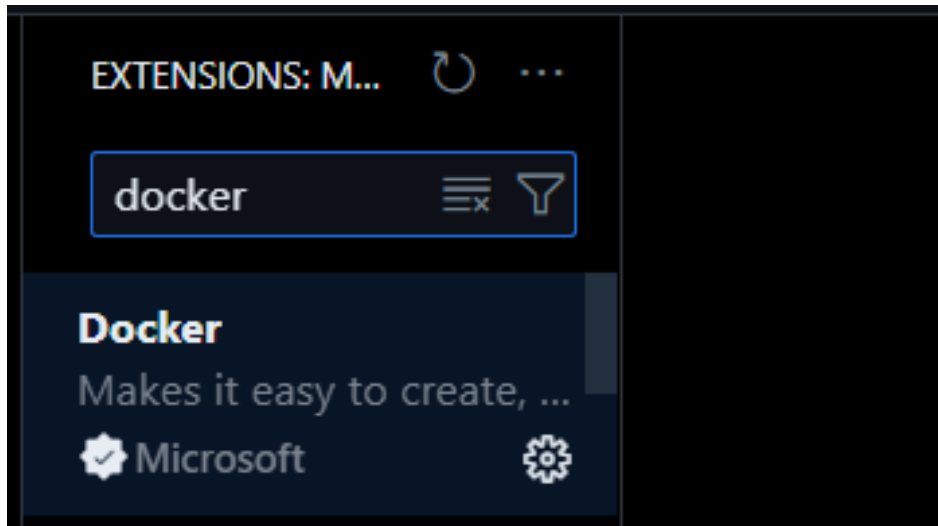


9. Now you can view all Git changes in your Source Control Tab.

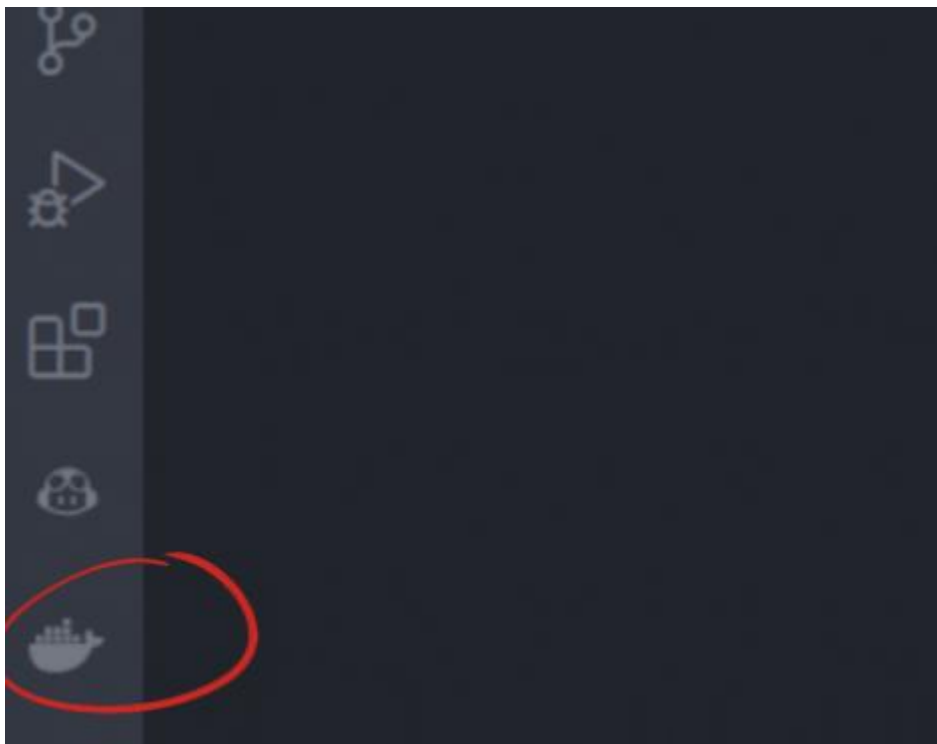


Docker in Visual Studio Code

1. Go to Extensions on VSCode and look up Docker, then click on the green Install button.

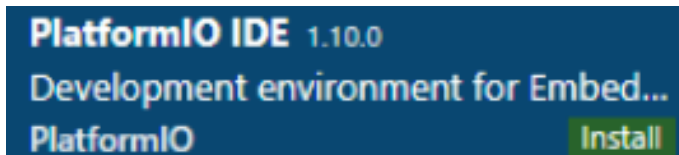


2. Once the extension is installed, click on its icon in VSCode. Here, you can see your images, registries, volumes, networks etc.



PlatformIO IDE

1. Open VS Code.
2. Click on the Extensions icon to open the Extensions tab.
3. Type PlatformIO IDE in the Search tab.
4. Click on the green Install button:



5. After that, the PlatformIO icon should show up on the left sidebar:

