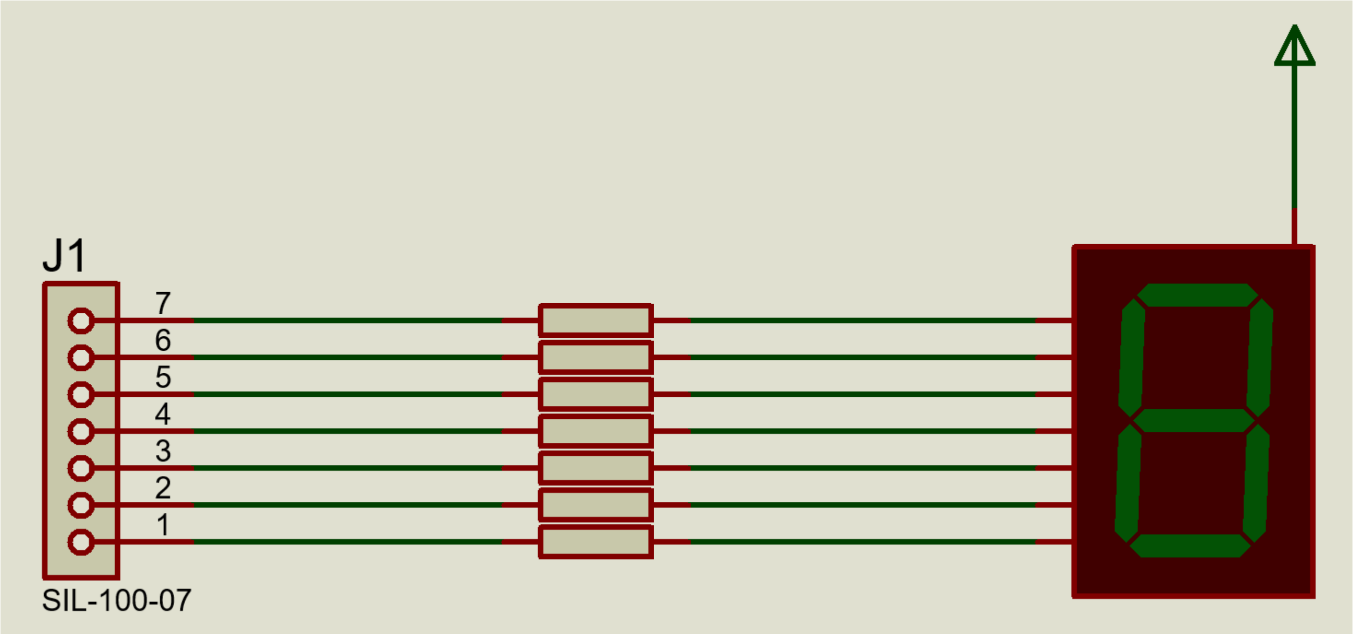
Pre Lab: Parallel Port

Group 1b2
E/20/157 – Janakantha S.M.B.G.
E/20/158 – Jananga T.G.C.
E/20/168 – Jayasinghe B.V.R.R.

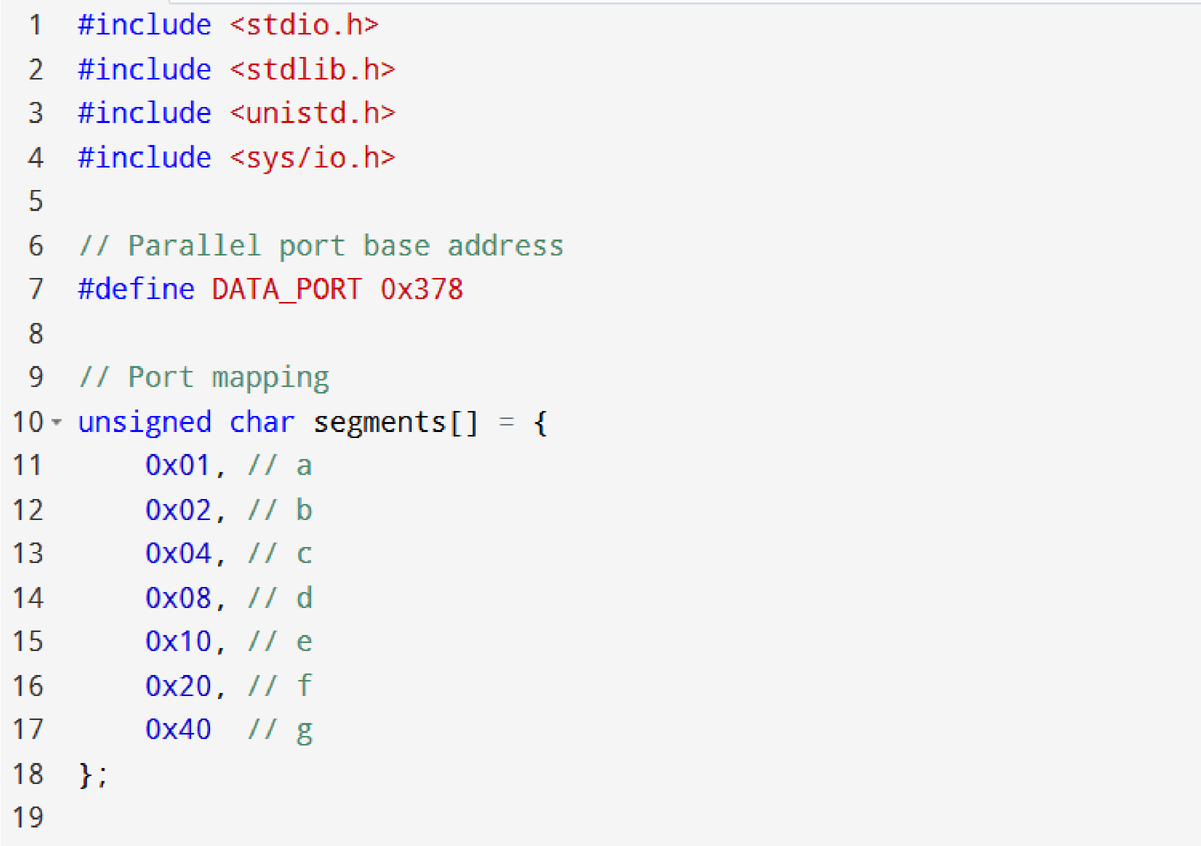
Part 01

1. Draw the circuit diagram that includes a 7-segment display and the data port of the parallel port. **Make sure that you have to connect separate resistors in series for each segment of SSD!**
   Calculate the resistance of the resistors that need to be connected.

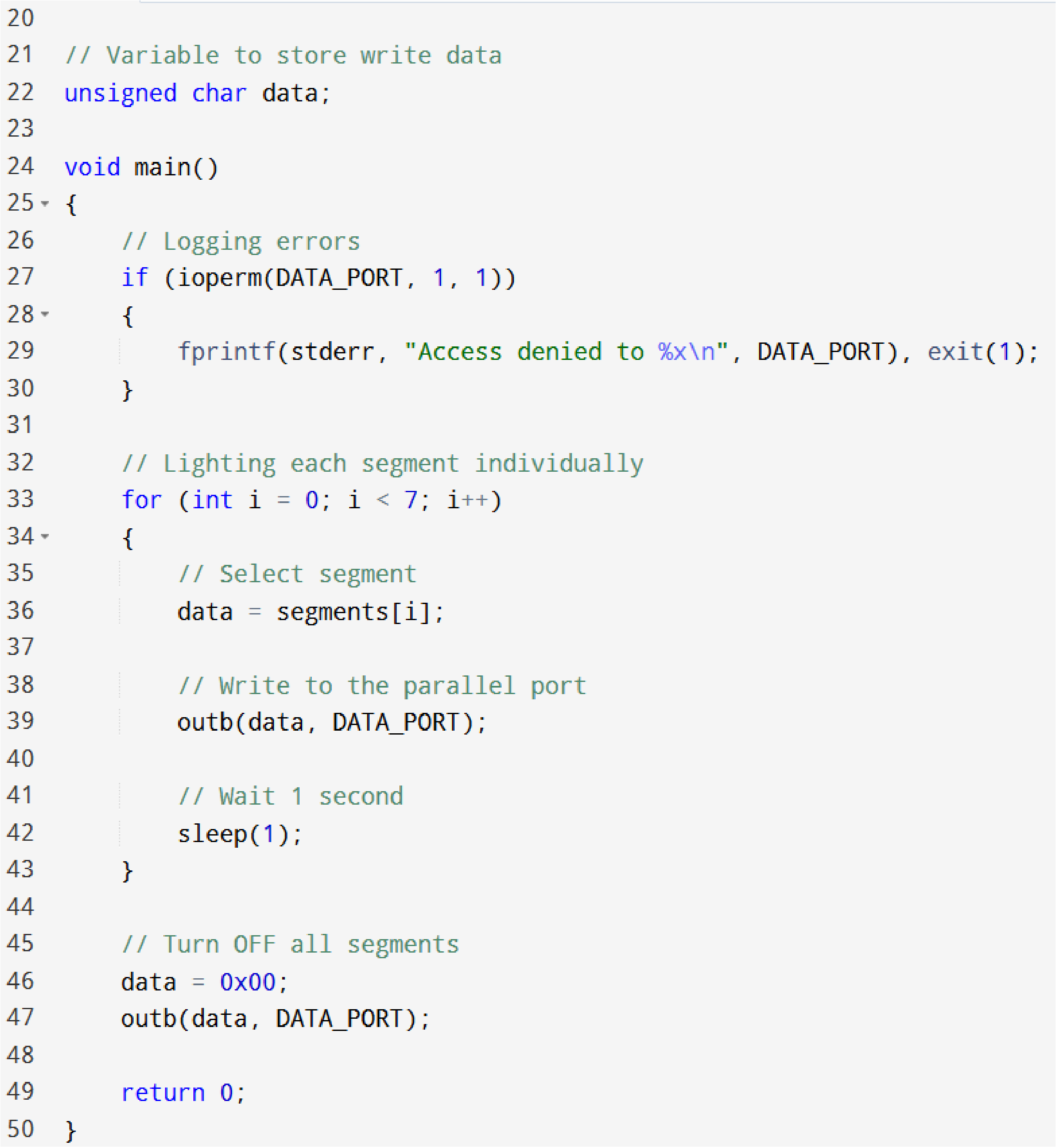


$V_{HIGH} = 5V, i_{max} = 20\ mA, R_{min} = 250\ \Omega$
$\therefore R \geq 250\ \Omega$

2. Write a program to light up each segment of the SSD one by one.



```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/io.h>

// Parallel port base address
#define DATA_PORT 0x378

// Port mapping
unsigned char segments[] = {
    0x01, // a
    0x02, // b
    0x04, // c
    0x08, // d
    0x10, // e
    0x20, // f
    0x40  // g
};

```



```c

// Variable to store write data
unsigned char data;

void main()
{
    // Logging errors
    if (ioperm(DATA_PORT, 1, 1))
    {
        fprintf(stderr, "Access denied to %x\n", DATA_PORT), exit(1);
    }

    // Lighting each segment individually
    for (int i = 0; i < 7; i++)
    {
        // Select segment
        data = segments[i];

        // Write to the parallel port
        outb(data, DATA_PORT);

        // Wait 1 second
        sleep(1);
    }

    // Turn OFF all segments
    data = 0x00;
    outb(data, DATA_PORT);

    return 0;
}
```

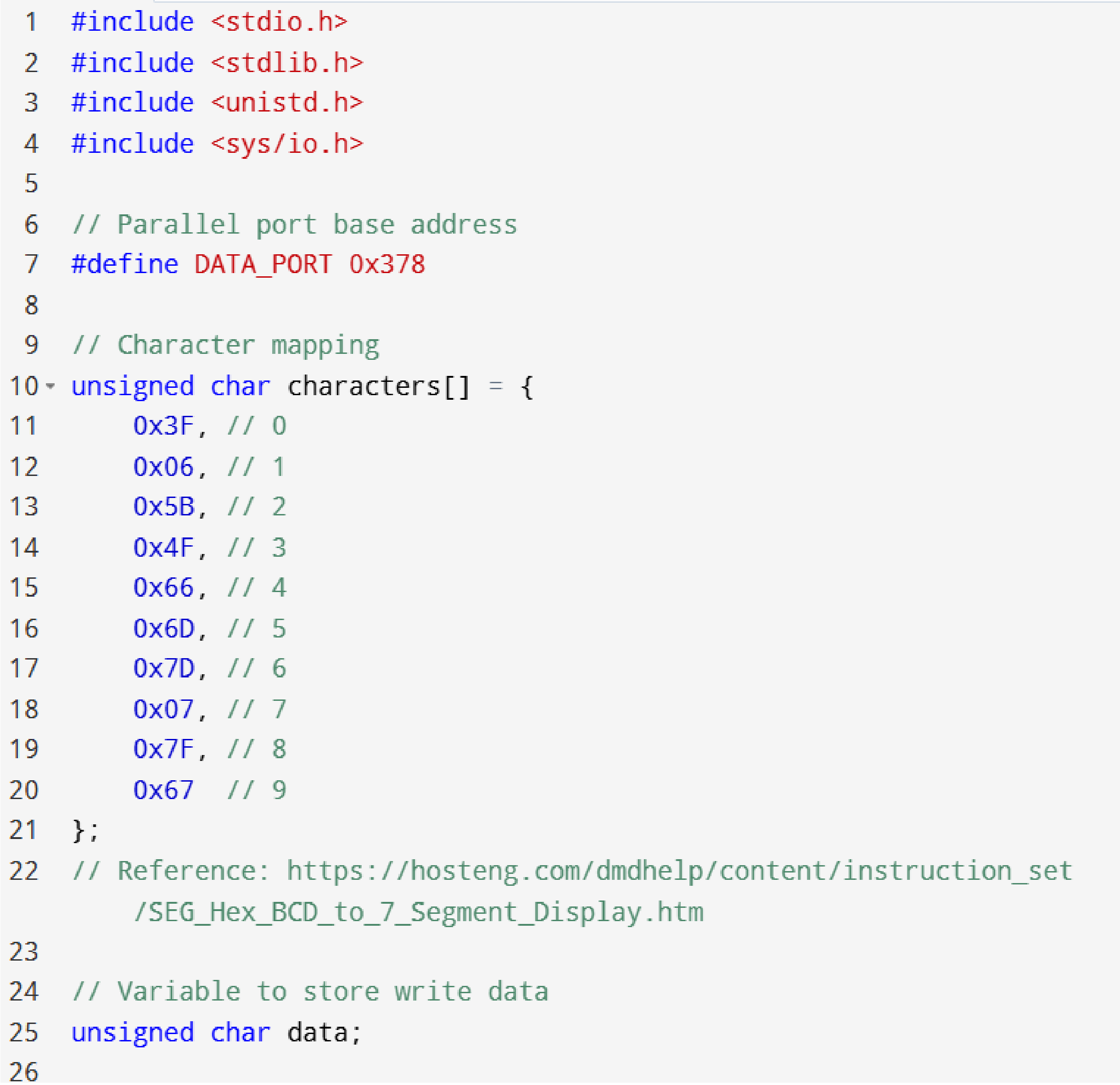
Part 2: Display 0-9 numbers on a single 7 segment display

1. Differentiate between the common anode and common cathode 7-segment display.

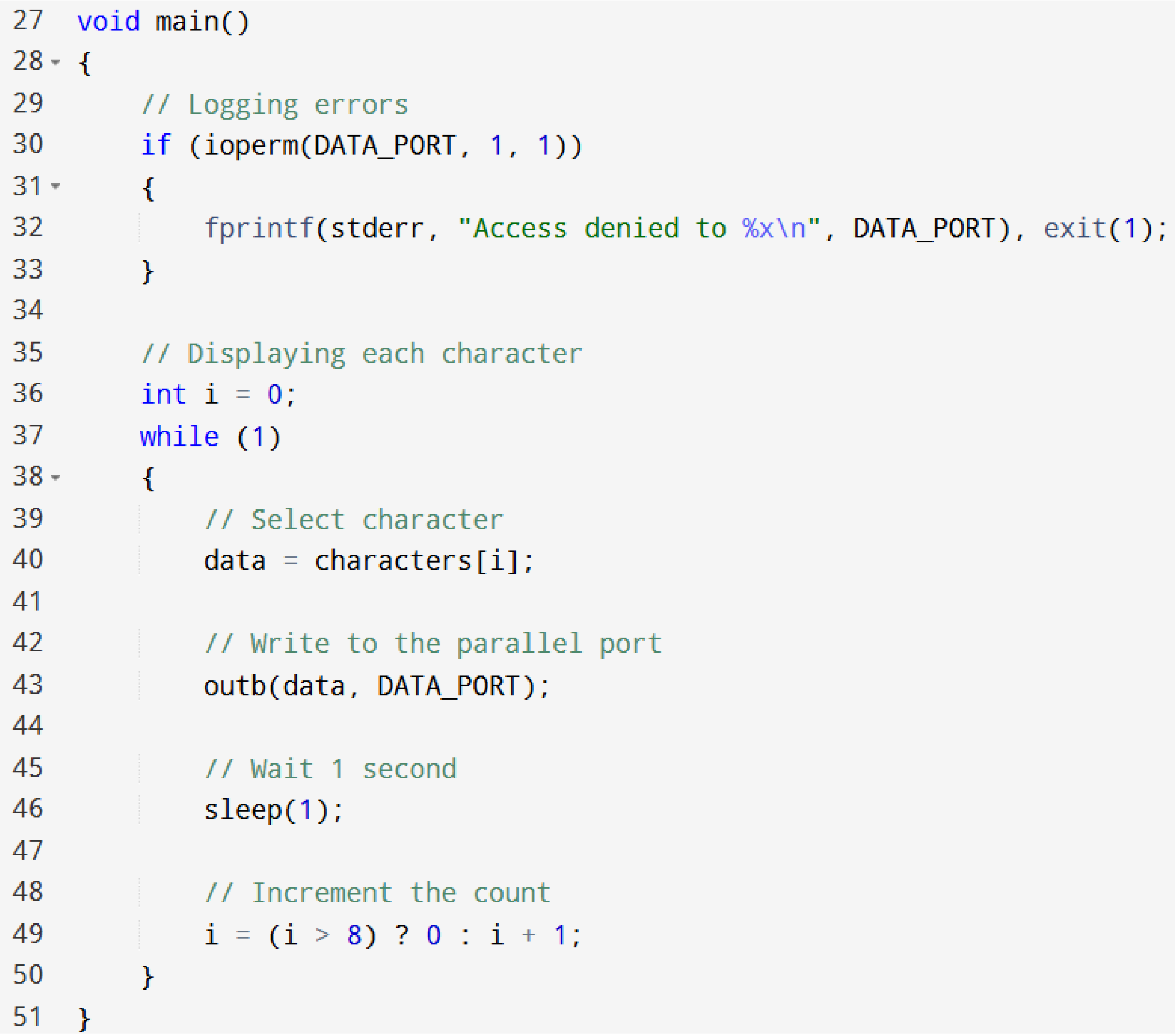
- Common Anode (CA):
    - All anodes of the LEDs are connected to +V.
    - A segment glows when its cathode is connected to LOW (0V).
    - Logic LOW = ON, HIGH = OFF.

- Common Cathode (CC):
    - All cathodes of LEDs are connected to GND.
    - A segment glows when its anode is connected to HIGH (+V).
    - Logic HIGH = ON, LOW = OFF.

2. Write a program to display characters from 0-9 in an infinite loop with a delay of 1 second between each character.



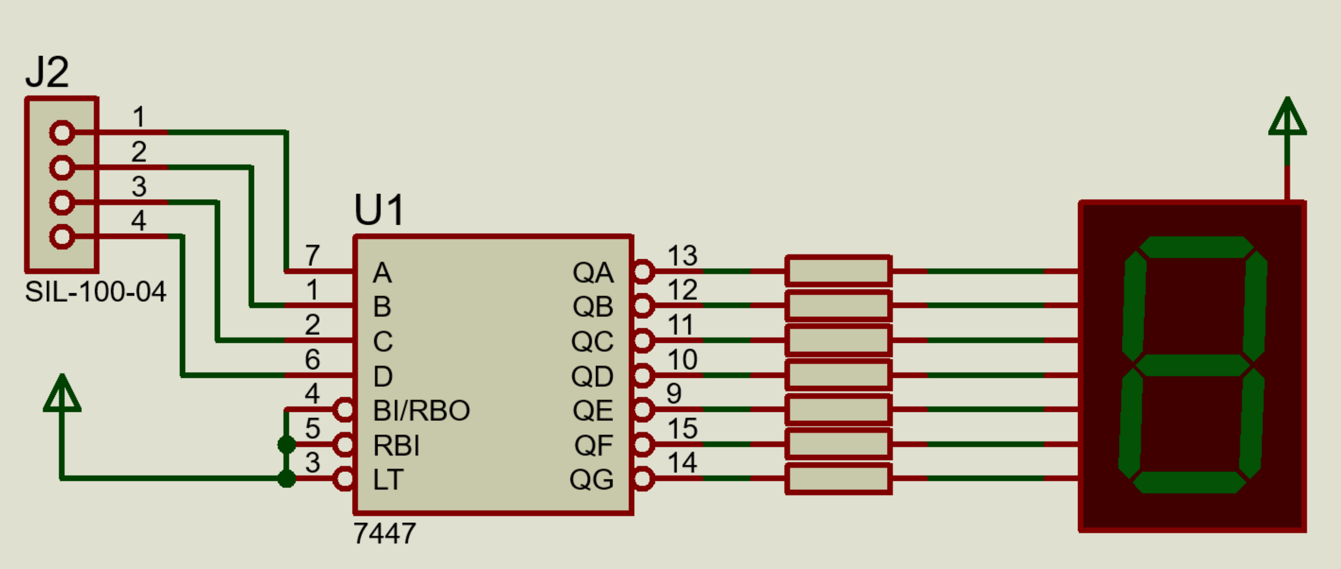
```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/io.h>

// Parallel port base address
#define DATA_PORT 0x378

// Character mapping
unsigned char characters[] = {
    0x3F, // 0
    0x06, // 1
    0x5B, // 2
    0x4F, // 3
    0x66, // 4
    0x6D, // 5
    0x7D, // 6
    0x07, // 7
    0x7F, // 8
    0x67  // 9
};
// Reference: https://hosteng.com/dmdhelp/content/instruction_set/SEG_Hex_BCD_to_7_Segment_Display.htm

// Variable to store write data
unsigned char data;

```



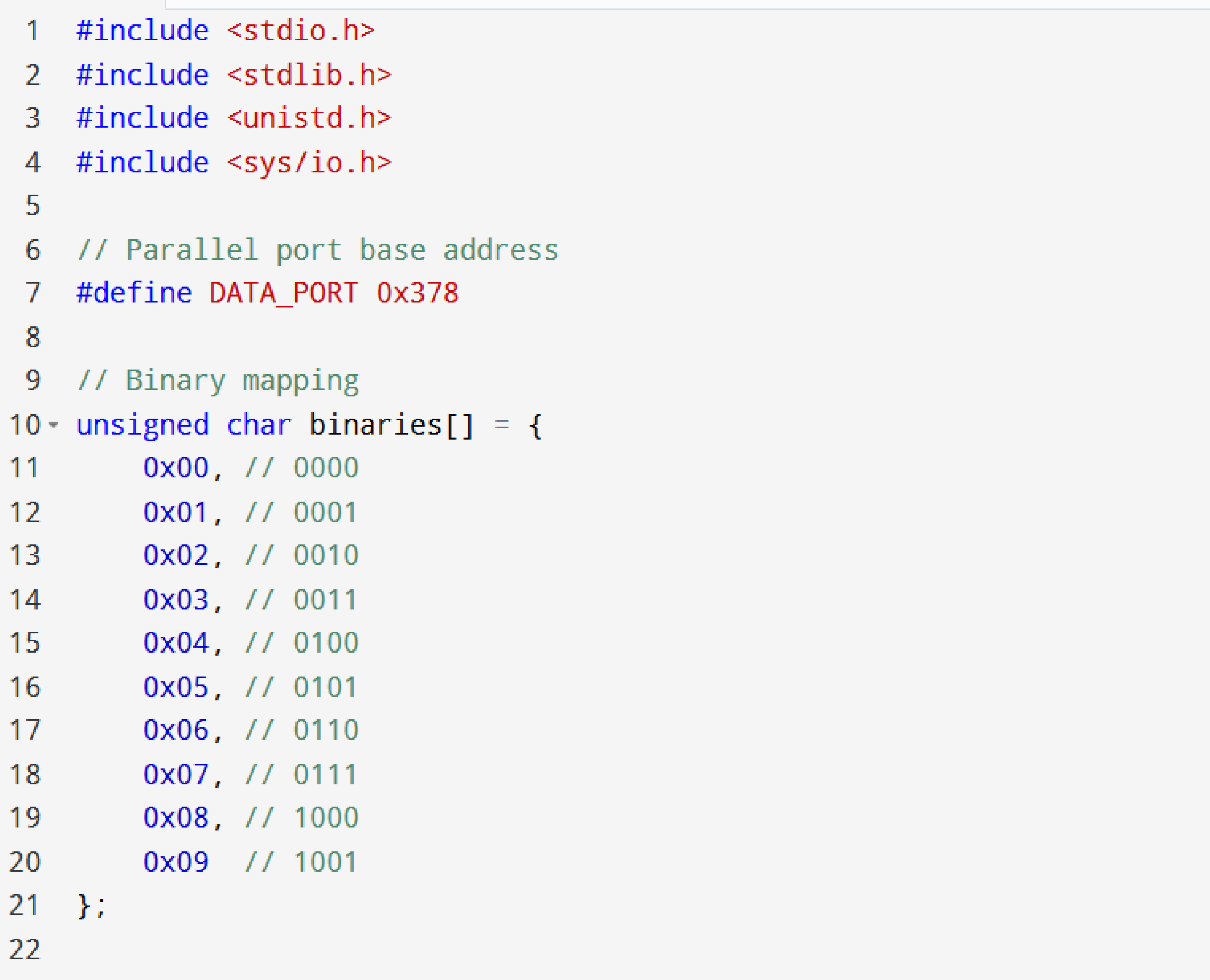
```c
void main()
{
    // Logging errors
    if (ioperm(DATA_PORT, 1, 1))
    {
        fprintf(stderr, "Access denied to %x\n", DATA_PORT), exit(1);
    }

    // Displaying each character
    int i = 0;
    while (1)
    {
        // Select character
        data = characters[i];

        // Write to the parallel port
        outb(data, DATA_PORT);

        // Wait 1 second
        sleep(1);

        // Increment the count
        i = (i > 8) ? 0 : i + 1;
    }
}
```

Part 3: Display 0-9 numbers on a single 7 segment display using 74LS47 IC

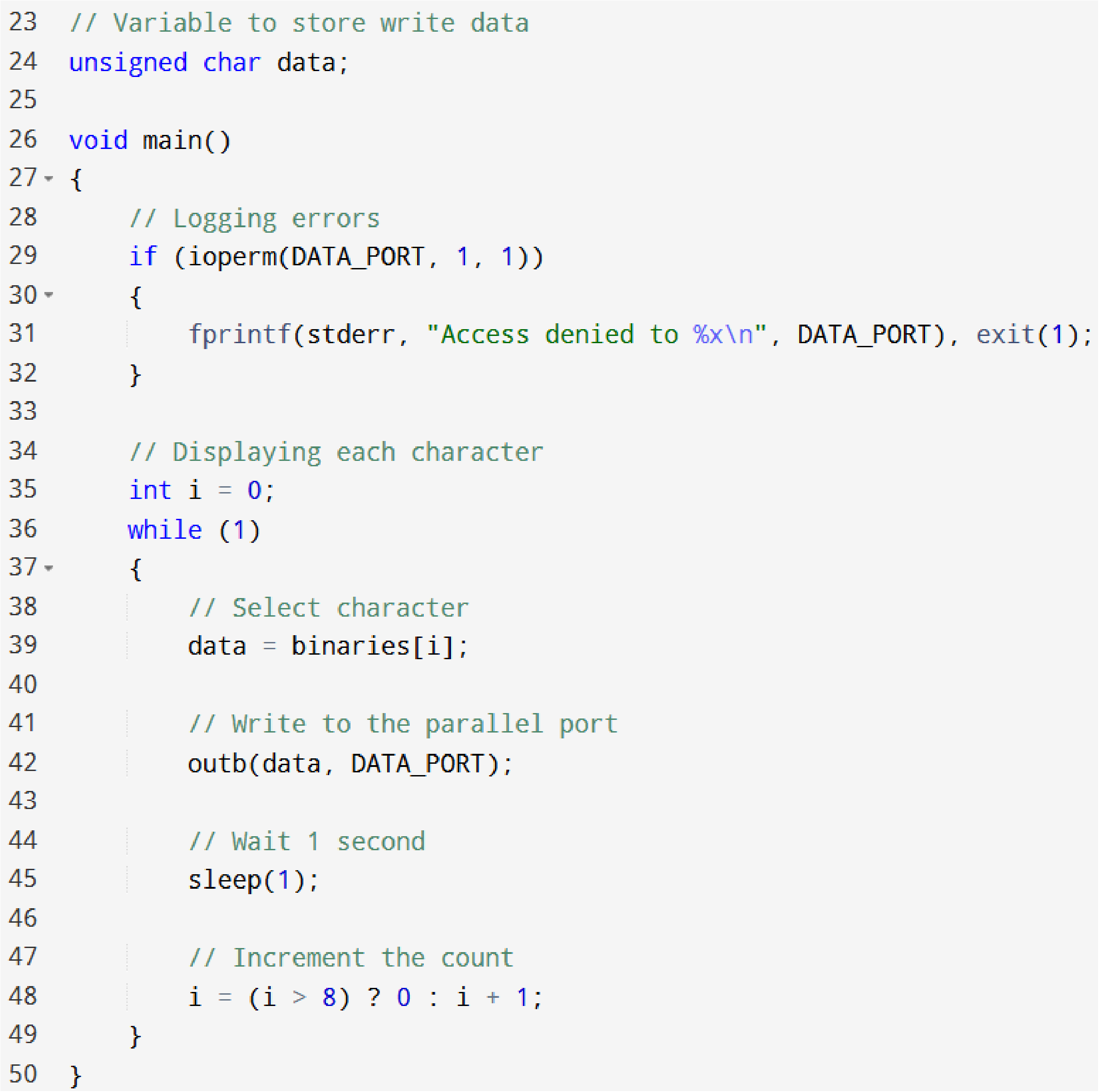
1. Draw the circuit diagram that includes a 7-segment display, 74LS47 IC, and the parallel port. Refer to the datasheet of the 74LS47 IC to find the least significant bit of the output. (Use a common anode)



2. Write the program to display characters from 0-9 in an infinite loop with a delay of 1 second between each character.



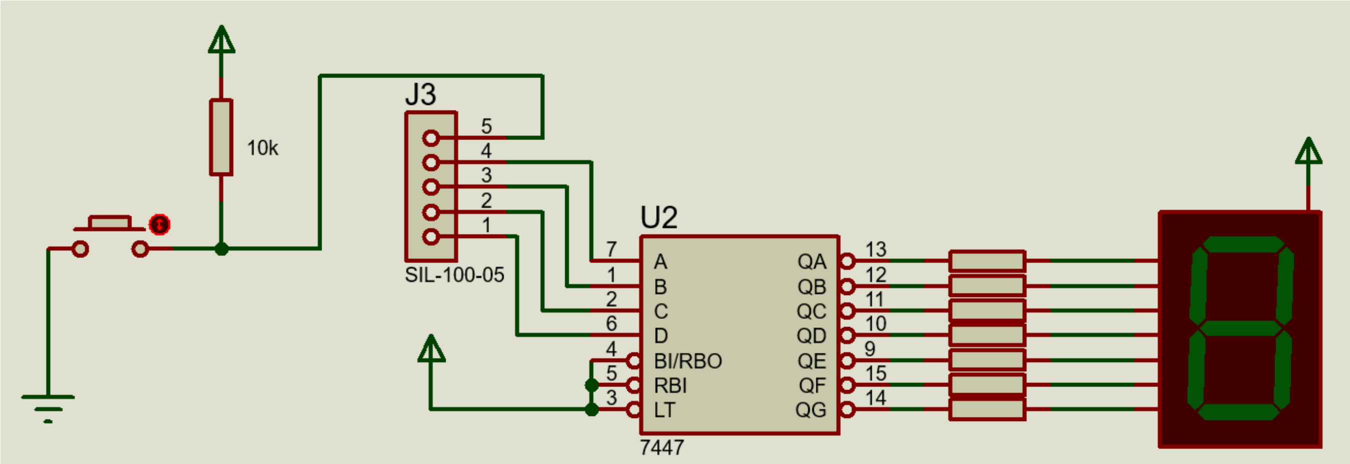
```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/io.h>

// Parallel port base address
#define DATA_PORT 0x378

// Binary mapping
unsigned char binaries[] = {
    0x00, // 0000
    0x01, // 0001
    0x02, // 0010
    0x03, // 0011
    0x04, // 0100
    0x05, // 0101
    0x06, // 0110
    0x07, // 0111
    0x08, // 1000
    0x09  // 1001
};

```

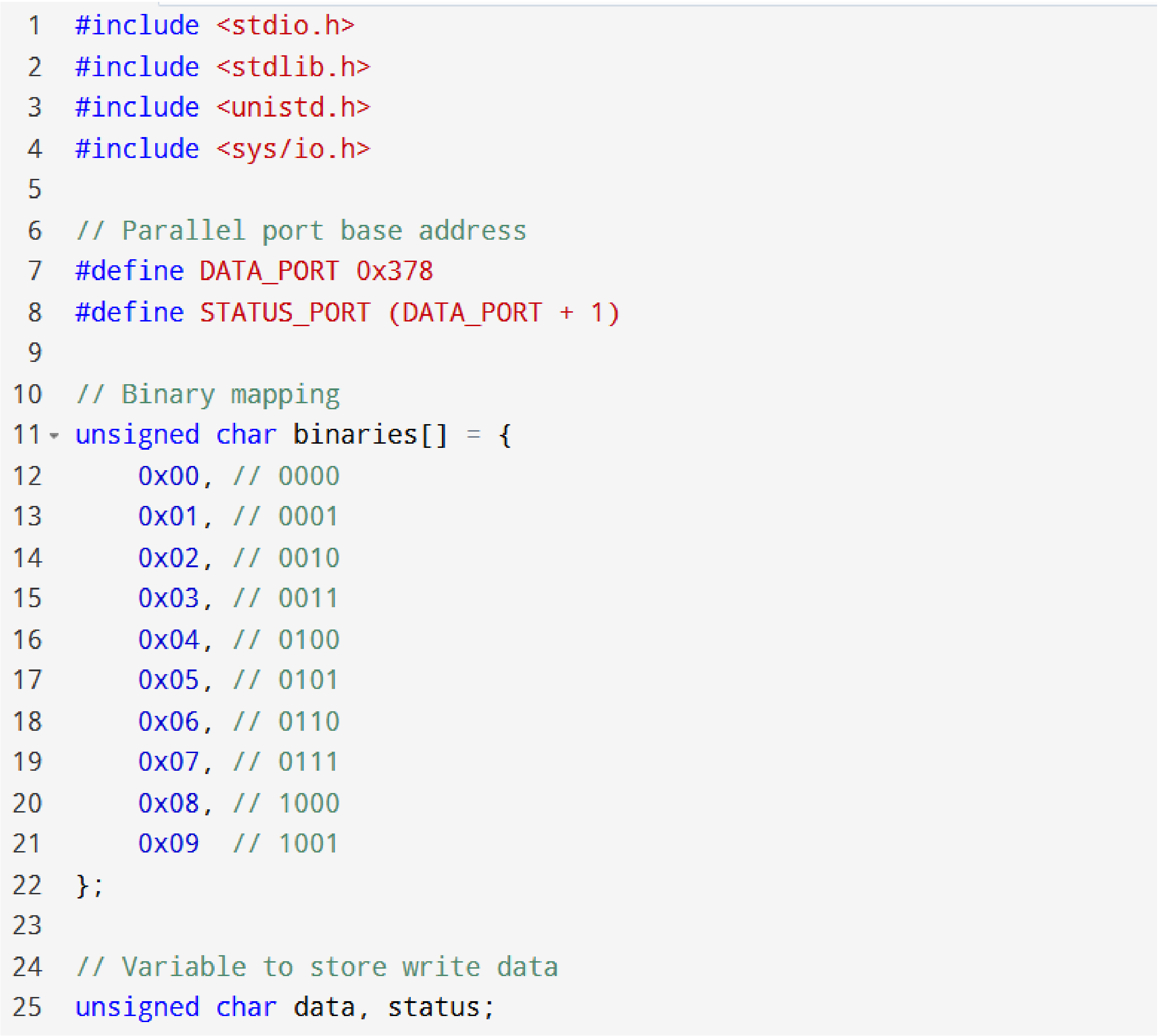


```c
// Variable to store write data
unsigned char data;

void main()
{
    // Logging errors
    if (ioperm(DATA_PORT, 1, 1))
    {
        fprintf(stderr, "Access denied to %x\n", DATA_PORT), exit(1);
    }

    // Displaying each character
    int i = 0;
    while (1)
    {
        // Select character
        data = binaries[i];

        // Write to the parallel port
        outb(data, DATA_PORT);

        // Wait 1 second
        sleep(1);

        // Increment the count
        i = (i > 8) ? 0 : i + 1;
    }
}
```

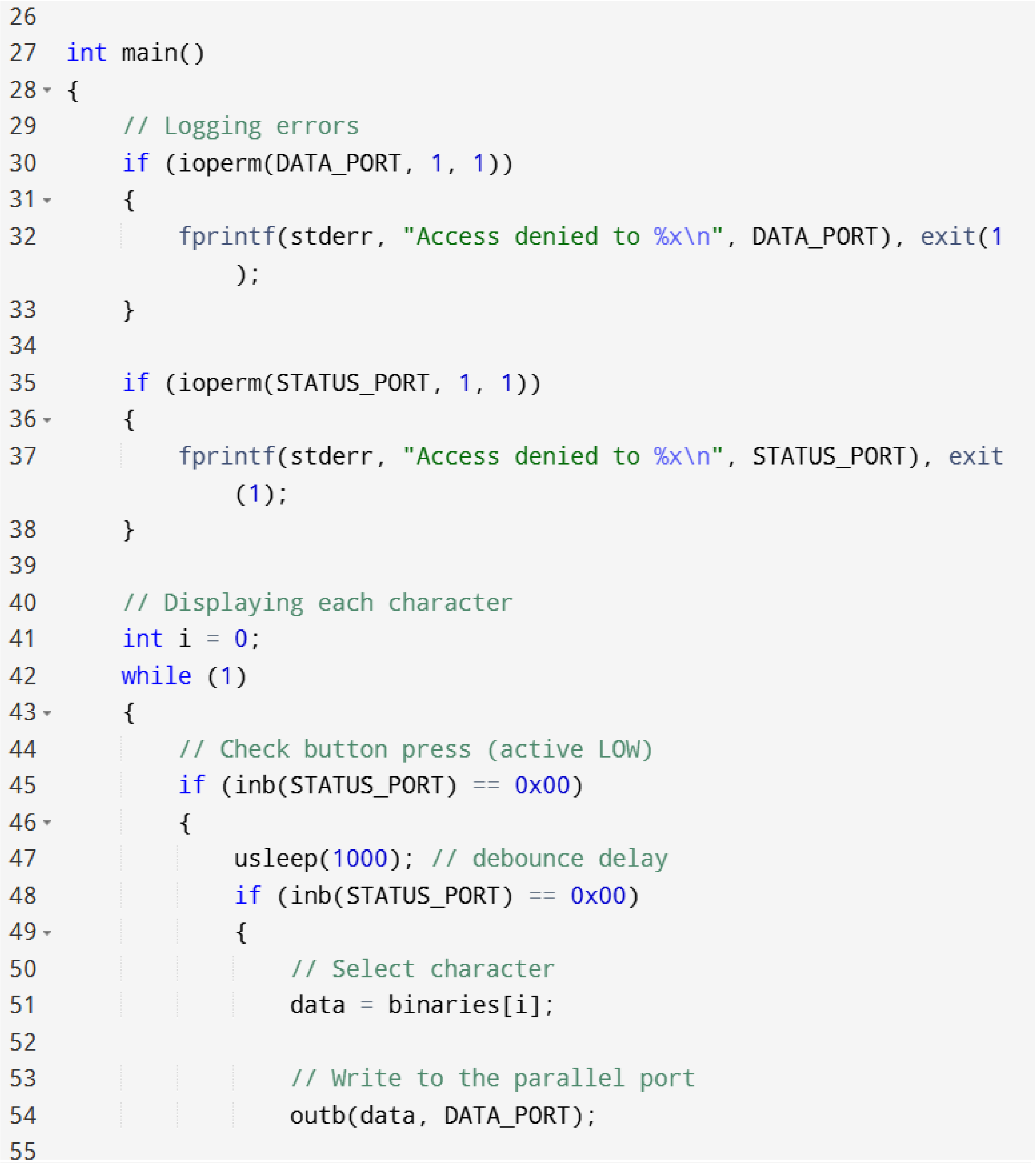
Part 4: Change the numbers displayed in the SSD with a push button

1. **Draw the circuit diagram** that includes a push button to take inputs, a 7-segment display, a **74LS47 IC** to show outputs and the parallel port. Make sure you use **proper resistors (pull-up/pull-down)** when taking inputs through the push button.

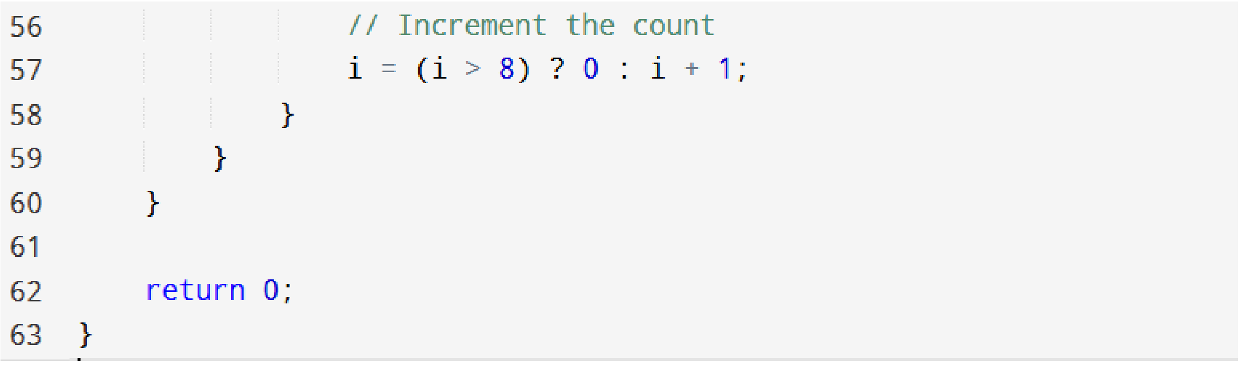




```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/io.h>

// Parallel port base address
#define DATA_PORT 0x378
#define STATUS_PORT (DATA_PORT + 1)

// Binary mapping
unsigned char binaries[] = {
    0x00, // 0000
    0x01, // 0001
    0x02, // 0010
    0x03, // 0011
    0x04, // 0100
    0x05, // 0101
    0x06, // 0110
    0x07, // 0111
    0x08, // 1000
    0x09  // 1001
};

// Variable to store write data
unsigned char data, status;
```



```c

int main()
{
    // Logging errors
    if (ioperm(DATA_PORT, 1, 1))
    {
        fprintf(stderr, "Access denied to %x\n", DATA_PORT), exit(1);
    }

    if (ioperm(STATUS_PORT, 1, 1))
    {
        fprintf(stderr, "Access denied to %x\n", STATUS_PORT), exit(1);
    }

    // Displaying each character
    int i = 0;
    while (1)
    {
        // Check button press (active LOW)
        if (inb(STATUS_PORT) == 0x00)
        {
            usleep(1000); // debounce delay
            if (inb(STATUS_PORT) == 0x00)
            {
                // Select character
                data = binaries[i];

                // Write to the parallel port
                outb(data, DATA_PORT);

```



```c
                // Increment the count
                i = (i > 8) ? 0 : i + 1;
            }
        }
    }

    return 0;
}
```