

Activity 1

1. **Text Segment:** Stores the executable code of the program, including all functions and instructions.
2. **Data Segment:** Stores the global variable size and the static variable result from the sum function.
3. **Heap Segment:** When line 12 is executed for the third time, the heap stores the first three user input values (e.g., [4, 5, 8, 0]).

Stack Segment: The stack stores local variables p, i, and total. At this point, i=2 and total contains the sum of the first three inputs.

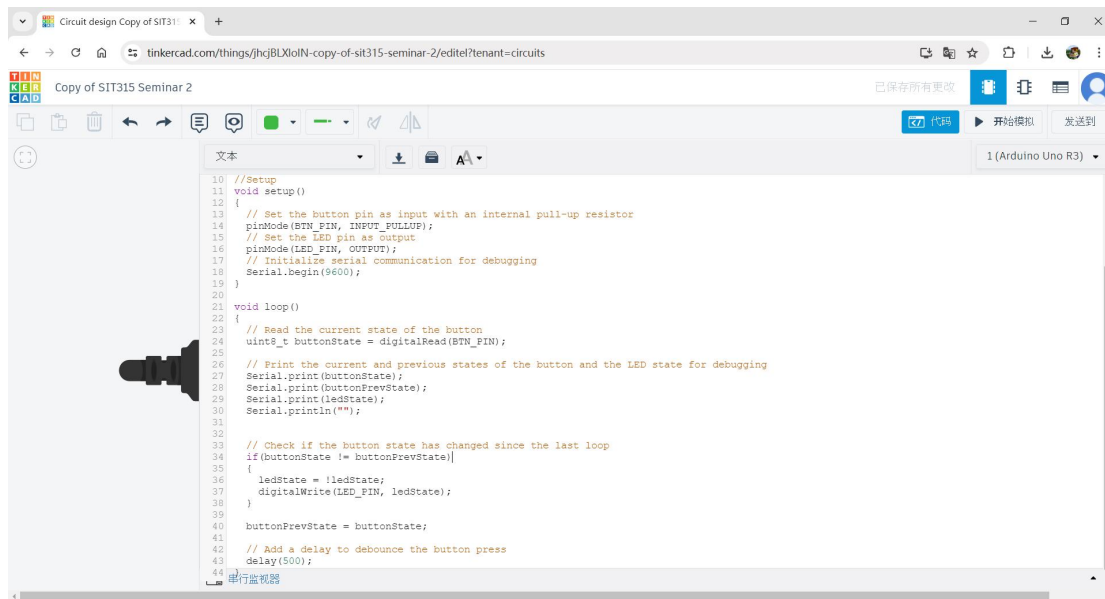
4. **Heap Segment:** When line 33 is executed, the dynamically allocated array on the heap has been released.

Stack Segment: The stack stores local variables p, i, and total. At this point, total contains the sum of all inputs, and i=4.

Activity 2

1.

```
1 // Define the pin numbers for the button and the LED
2 const uint8_t BTN_PIN = 2;
3 const uint8_t LED_PIN = 13;
4
5 // Initialize variables to store the previous state of the button and the current state of the LED
6 uint8_t buttonPrevState = LOW;
7 uint8_t ledState = LOW;
8
9
10 // Setup
11 void setup()
12 {
13 // Set the button pin as input with an internal pull-up resistor
14 pinMode(BTN_PIN, INPUT_PULLUP);
15 // Set the LED pin as output
16 pinMode(LED_PIN, OUTPUT);
17 // Initialize serial communication for debugging
18 Serial.begin(9600);
19 }
20
21 void loop()
22 {
23 // Read the current state of the button
24 uint8_t buttonState = digitalRead(BTN_PIN);
25
26 // Print the current and previous states of the button and the LED state for debugging
27 Serial.print(buttonState);
28 Serial.print(buttonPrevState);
29 Serial.print(ledState);
30 Serial.println("");
31
32 // Check if the button state has changed since the last loop
33 if (buttonState != buttonPrevState)
34 {
35 // Toggle the LED state
36 ledState = !ledState;
37 digitalWrite(LED_PIN, ledState);
38 }
39 buttonPrevState = buttonState;
40 }
```



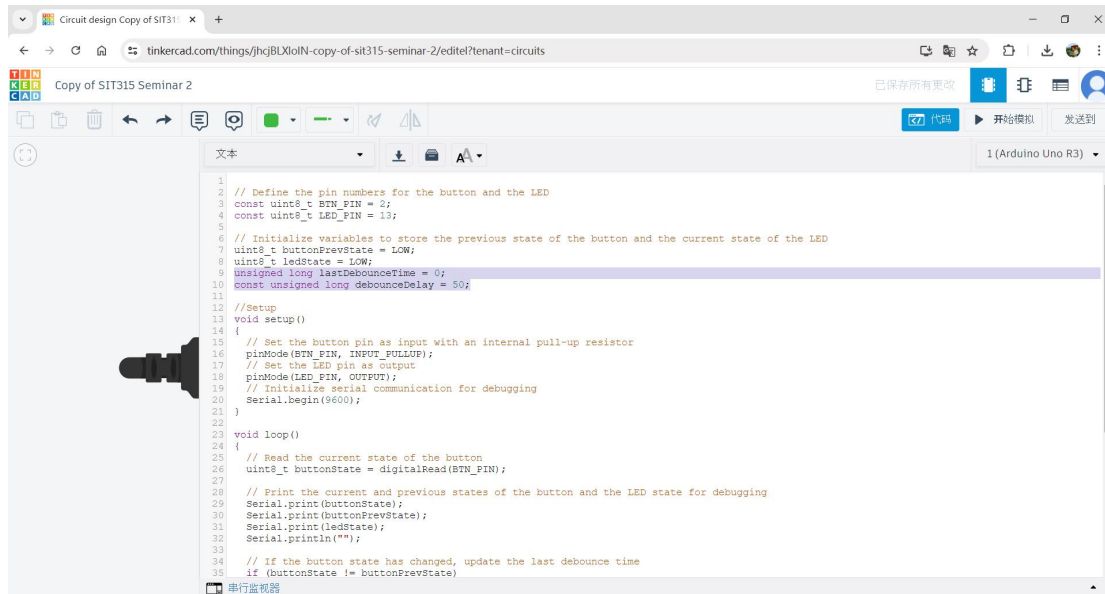
The screenshot shows the Tinkercad code editor with a circuit diagram on the left and a code editor on the right. The code is for an Arduino Uno R3 and implements a simple button press to toggle an LED. It uses a pull-up resistor for the button and a delay for debouncing.

```
10 //Setup
11 void setup()
12 {
13   // Set the button pin as input with an internal pull-up resistor
14   pinMode(BTN_PIN, INPUT_PULLUP);
15   // Set the LED pin as output
16   pinMode(LED_PIN, OUTPUT);
17   // Initialize serial communication for debugging
18   Serial.begin(9600);
19 }
20
21 void loop()
22 {
23   // Read the current state of the button
24   uint8_t buttonState = digitalRead(BTN_PIN);
25
26   // Print the current and previous states of the button and the LED state for debugging
27   Serial.print(buttonState);
28   Serial.print(buttonPrevState);
29   Serial.print(ledState);
30   Serial.println("");
31
32   // Check if the button state has changed since the last loop
33   if (buttonState != buttonPrevState)
34   {
35     ledState = !ledState;
36     digitalWrite(LED_PIN, ledState);
37   }
38   buttonPrevState = buttonState;
39
40   // Add a delay to debounce the button press
41   delay(500);
42 }
```

2.

The main problem with the code is the lack of proper debouncing for the button press, which could result in multiple toggles of the LED for a single button press due to the mechanical noise of the button.

3.



The screenshot shows the Tinkercad code editor with a circuit diagram on the left and a code editor on the right. The code is for an Arduino Uno R3 and implements a button press to toggle an LED. It uses a pull-up resistor for the button and a delay for debouncing. The code is more detailed than the previous one, including comments for each step.

```
1 // Define the pin numbers for the button and the LED
2 const uint8_t BTN_PIN = 2;
3 const uint8_t LED_PIN = 13;
4
5 // Initialize variables to store the previous state of the button and the current state of the LED
6 uint8_t buttonPrevState = LOW;
7 uint8_t ledState = LOW;
8 unsigned long lastDebounceTime = 0;
9 const unsigned long debounceDelay = 50;
10
11 //Setup
12 void setup()
13 {
14   // Set the button pin as input with an internal pull-up resistor
15   pinMode(BTN_PIN, INPUT_PULLUP);
16   // Set the LED pin as output
17   pinMode(LED_PIN, OUTPUT);
18   // Initialize serial communication for debugging
19   Serial.begin(9600);
20 }
21
22 void loop()
23 {
24   // Read the current state of the button
25   uint8_t buttonState = digitalRead(BTN_PIN);
26
27   // Print the current and previous states of the button and the LED state for debugging
28   Serial.print(buttonState);
29   Serial.print(buttonPrevState);
30   Serial.print(ledState);
31   Serial.println("");
32
33   // If the button state has changed, update the last debounce time
34   if (buttonState != buttonPrevState)
35   {
36     lastDebounceTime = millis();
37     ledState = !ledState;
38     digitalWrite(LED_PIN, ledState);
39     buttonPrevState = buttonState;
40   }
41
42   // Check if the button has been pressed for a long enough time to toggle the LED
43   if (millis() - lastDebounceTime > debounceDelay)
44   {
45     ledState = !ledState;
46     digitalWrite(LED_PIN, ledState);
47   }
48 }
```

Circuit design Copy of SIT315

tinkercad.com/things/hqjBLXIoIN-copy-of-sit315-seminar-2/edit?tenant=circuits

Copy of SIT315 Seminar 2

已保存所有更改

代码

开始模拟

发送到

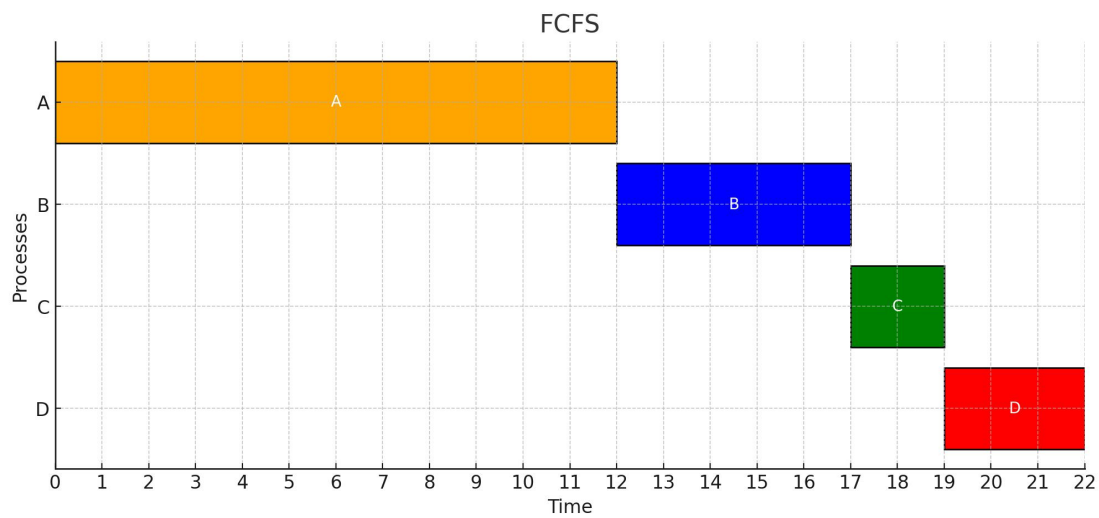
1 (Arduino Uno R3)

```
20 Serial.begin(9600);
21 }
22
23 void loop()
24 {
25   // Read the current state of the button
26   uint8_t buttonState = digitalRead(BTN_PIN);
27
28   // Print the current and previous states of the button and the LED state for debugging
29   Serial.print(buttonState);
30   Serial.print(buttonPrevState);
31   Serial.print(ledState);
32   Serial.println("");
33
34   // If the button state has changed, update the last debounce time
35   if (buttonState != buttonPrevState)
36   {
37     lastDebounceTime = millis();
38   }
39
40   // Check if the button state has been stable for a period longer than the debounce delay
41   if ((millis() - lastDebounceTime) > debounceDelay)
42   {
43     if (buttonState != buttonPrevState) // Check if the button state has changed since the last loop
44     {
45       ledState = !ledState;
46       digitalWrite(LED_PIN, ledState);
47     }
48     buttonPrevState = buttonState;
49
50     // Add a delay to debounce the button press
51     delay(500);
52   }
53 }
54
```

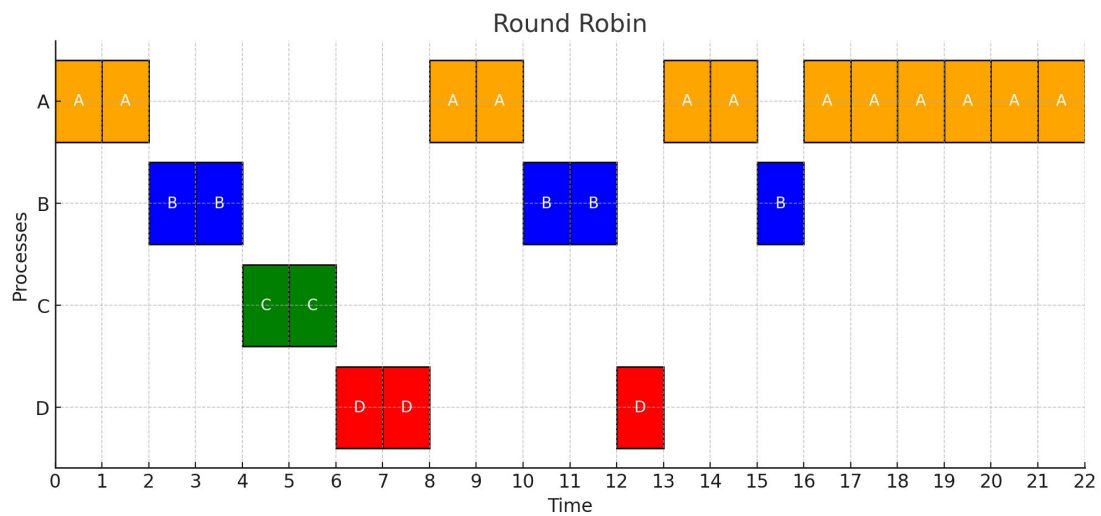
串行监视器

Activity 3

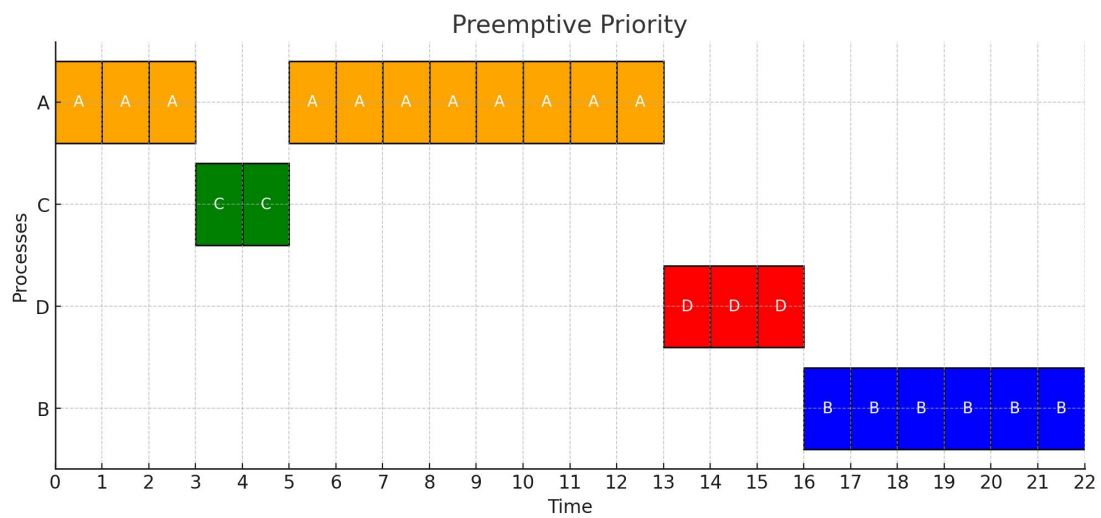
FCFS



Round-Robin



Preemptive Priority-Based



Waiting Time	A	B	C	D	Total	Average
FCFS	0	11	14	15	40	10
RR	10	10	1	6	27	6.75
PP	2	16	0	10	28	7

A scheduling algorithm with a lower waiting time is the Shortest Remaining Time First (SRTF). SRTF always selects the process with the shortest remaining burst time to execute next, minimizing the average waiting time.