Activity 1: Process Memory Segments

1. Identify what is stored in the text segment when the process is admitted by the OS and is in the ready state.
   - Text segment: The text segment contains the compiled machine code of the program's executable instructions.
2. Identify what variables will be stored in the data segment.
   - Size: A global variable with the value 4.
   - p: A pointer variable to int
   - i: An integer variable used as a loop counter in the main function.
   - total: An integer variable to store the cumulative sum.
3. Identify what is stored in heap and stack segments when line 12 is being executed for the third time.
   - Heap: After the third execution of line 12, the heap segment will store three dynamically allocated integers since the loop iterates three times.
   - Stack: The stack segment will contain local variables of the sum function during the third execution of line 12, such as n and result.
4. Identify what is stored in heap and stack segments when line 33 is being executed.
   - Heap: After line 33, the heap segment will be freed of the memory allocated for the array p using delete [] p.
   - Stack: The stack segment will still contain local variables of the main function, such as "p", "I", and "total".

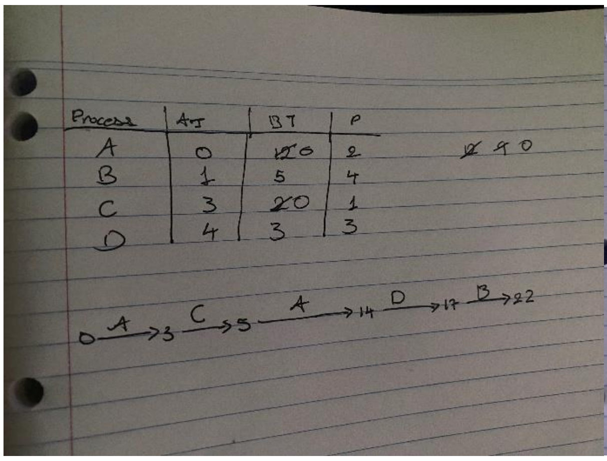
Activity 2: Interrupts

1. Based on the design and the code, explain what the primary function of this board is. Complete the code by adding appropriate comments in the designated lines.
   - The primary function of this board is to turn on the LED when the button is pressed. Essentially if the user holds the button, the LED light will stay on, other wise the light turns off.
2. Identify what the main problem in the code is and how it can affect the end-users.
   - The main problem in the code is that the button is not being activated in real-time; for example, the implementation of the interaction between the button and the background process is limited to the Arduino's firmware. The interaction of the button in the code is implemented in the loop cycle which has a delay of 500ms. This prevents the immediate input of pressing the button therefore the effect the of the light turning on is slowed down by 500ms as it loops constantly throughout the serial input.
3. Change the code to resolve the problem you identified in 2.

```
//ToDo Add Comment
const uint8_t BTN_PIN = 2;
const uint8_t LED_PIN = 13;

//ToDo Add Comment
uint8_t buttonPrevState = LOW;
uint8_t ledState = LOW;

//ToDo Add Comment
void setup()
{
  //ToDo Add Comment
  pinMode(BTN_PIN, INPUT_PULLUP);
  //ToDo Add Comment
  pinMode(LED_PIN, OUTPUT);
  //ToDo Add Comment
  Serial.begin(9600);
  attachInterrupt(digitalPinToInterrupt(BTN_PIN),interrupt, CHANGE)
}

void loop()
{
}

void interrupt()
{
  ledState = !ledState;
   digitalWrite(LED_PIN, ledState);
}
```

Activity 3- Process Scheduling

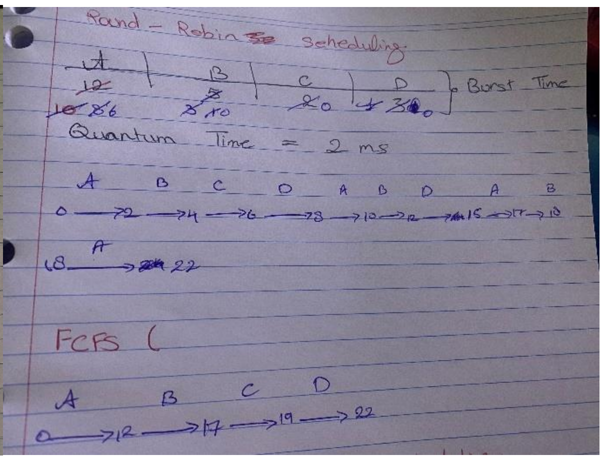
1. Draw three Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, Round-Robin, and Pre-emptive Priority-based scheduling.



Process	AT	BT	P
A	0	~~12~~10	2
B	1	5	4
C	3	20	1
D	4	3	3

~~2~~ 9 0

$0 \xrightarrow{A} 3 \xrightarrow{C} 5 \xrightarrow{A} 14 \xrightarrow{D} 17 \xrightarrow{B} 22$



Round - Robin ~~Sc~~ Scheduling

A	B	C	D
~~12~~	~~5~~		
~~10~~ ~~8~~ 6	~~3~~ 10	~~2~~0	~~4~~ 3~~1~~0

} Burst Time

Quantum Time = 2 ms

$0 \xrightarrow{A} 2 \xrightarrow{B} 4 \xrightarrow{C} 6 \xrightarrow{D} 8 \xrightarrow{A} 10 \xrightarrow{B} 12 \xrightarrow{D} 15 \xrightarrow{A} 17 \xrightarrow{B} 18$

$18 \xrightarrow{A} 22$

FCFS (

$0 \xrightarrow{A} 12 \xrightarrow{B} 17 \xrightarrow{C} 19 \xrightarrow{D} 22$

2. For each of the scheduling algorithms, compute the waiting times of each process.
   - FCFS: A = 0, B = 12 – 1 = 11, C = 17 – 3 = 14, D = 19 – 4 = 15.
   - Round Robin:

Process	Completion Time	Arrival Time	Turn Around Time	Burst Time	Waiting Time
A	22	0	22 – 0 = 22	12	22 – 12 = 10
B	18	1	18 – 1 = 17	5	17 – 5 = 12
C	6	3	6 – 3 = 3	2	3 – 2 = 1
D	15	4	15 – 4 = 11	3	11 – 3 = 8

   - Preemptive Priority Based Scheduling:

Process	Completion Time	Arrival Time	Turn Around Time	Burst Time	Waiting Time
A	14	0	14 – 0 = 14	12	14 – 12 = 2
B	22	1	22 – 1 = 21	5	22 – 5 = 17
C	5	3	5 – 3 = 2	2	5 – 2 = 3
D	17	4	17 – 4 = 13	3	17 – 3 = 14

3. Average Waiting Time
   - FCFS: (0 + 11 + 14 + 15) / 4 = 19
   - Round Robin: (10 + 12 + 1 + 8) / 4 = 7.75
   - Preemptive Priority: (2 + 17 + 3 + 14) / 4 = 9
4. Research and find another scheduling algorithm that has a lower waiting time than these three algorithms.
   - The Shortest Remaining Time (SRT) scheduling algorithm is a preemptive approach that selects the process with the smallest remaining burst time to execute next. It aims to minimize waiting times by prioritizing shorter jobs. However, SRT requires knowing burst times in advance, faces context switch overhead, and may cause starvation for longer processes.
   - Wait Time for each process:

Process	Completion Time	Arrival Time	Turn Around Time	Burst Time	Waiting Time
A	22	0	22 – 0 = 22	12	22 – 12 = 2
B	11	1	11 – 1 = 10	5	11 – 5 = 17
C	5	3	5 – 3 = 2	2	5 – 2 = 3
D	8	4	8 – 4 = 4	3	8 – 3 = 14

   - Average Wait Time: (10 + 6 + 3 + 5) / 4 = 6