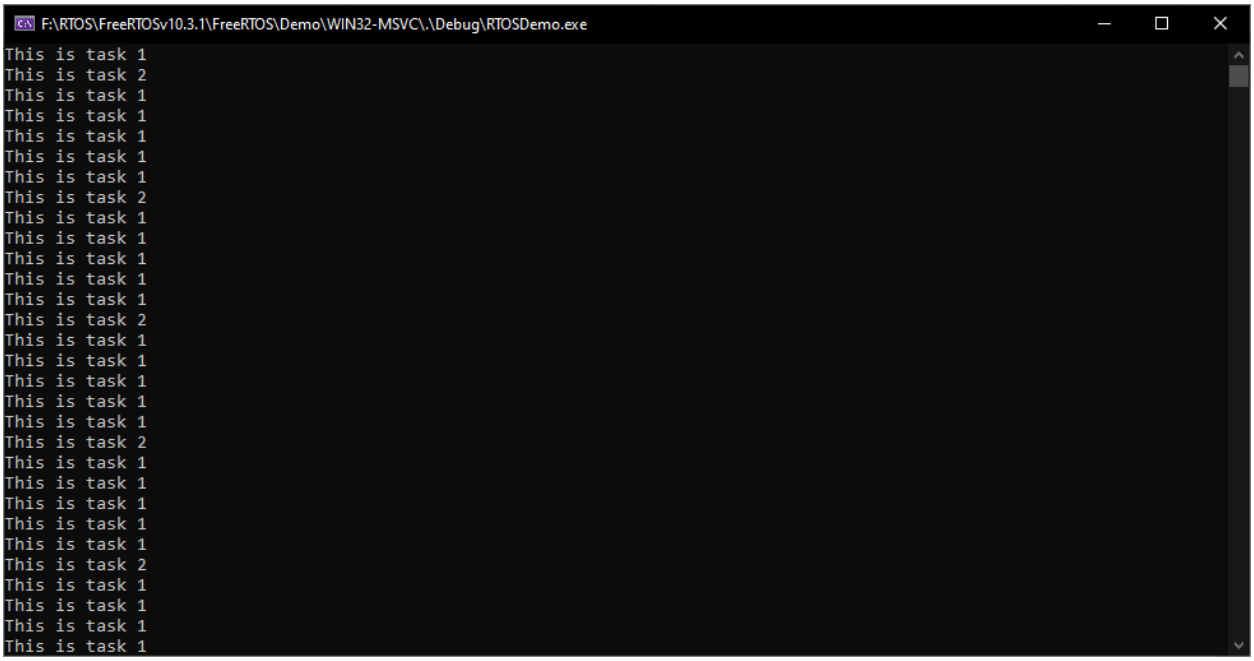
Assignment 1

Output:



Task 1:
name="Task1"
stack size = 1000
priority = 3
Task1 should print out "This is task 1" every 100 milliseconds

Task 2:
name="Task2"
stack size = 100
priority = 1
Task1 should print out "This is task 2" every 500 milliseconds

- Since Task 2 is of a higher priority, we see that it's printed instead of Task 1 on the second instance.
- After Every 5 Task 1 execution, we see Task 2 because the period of execution is 100 and 500, respectively.
- I have used FreeRTOS Kernal V 10.3.1

Code:

```
142  void Task1()
143  {
144      while (1)
145      {
146          printf("This is task 1 \n"); // Content to be printed for Task 1
147          fflush(stdout);
148          vTaskDelay(100);             // Set a delay of 100mS
149
150      }
151  }
152
153   //Task 2--------------------------------------------------------
154
155  void Task2()
156  {
157      while (1)
158      {
159          printf("This is task 2 \n");  // Content to be printed for Task 2
160          fflush(stdout);
161          vTaskDelay(500);              // Set a delay of 500mS
162
163      }
164  }
165
166
167
168   /*-----------------------------------------------------------*/
```

```
/*-----------------------------------------------------------*/
int main( void )
{
    /* This demo uses heap_5.c, so start by defining some heap regions.  heap_5
    is only used for test and example reasons.  Heap_4 is more appropriate.  See
    http://www.freertos.org/a00111.html for an explanation. */
    prvInitialiseHeap();

    /* Initialise the trace recorder.  Use of the trace recorder is optional.
    See http://www.FreeRTOS.org/trace for more information. */
    vTraceEnable( TRC_START );


    xTaskHandle TaskHandler;                                // Created a Task Handler by the name TaskHandler

    xTaskCreate(Task1, "Task1", 1000, NULL, 3, &TaskHandler);   // Stack size 1000 and Priority 3
    xTaskCreate(Task2, "Task2", 100, NULL, 1, &TaskHandler);    // Stack size 100 and Priority 1
    vTaskStartScheduler();                                  // Initilized the Scheduler

    for (;;);

    return 0;
}

/*-----------------------------------------------------------*/
```