```
#include<Arduino_FreeRTOS.h>

#include<task.h>

#include<queue.h>


// Settings

static const uint8_t buf_len = 255;


// Globals

static char *msg_ptr = NULL;

static volatile uint8_t msg_flag = 0;


//*****************************************************************************

// Tasks


// Task: read message from Serial buffer

void readSerial(void *parameters) {


  char c;

  char buf[buf_len];

  uint8_t idx = 0;


  // Clear whole buffer

  memset(buf, 0, buf_len);


  // Loop forever

  while (1) {
```

```
// Read cahracters from serial
if (Serial.available() > 0) {
  c = Serial.read();

  // Store received character to buffer if not over buffer limit
  if (idx < buf_len - 1) {
    buf[idx] = c;
    idx++;
  }

  // Create a message buffer for print task
  if (c == '\n') {

    // The last character in the string is '\n', so we need to replace
    // it with '\0' to make it null-terminated
    buf[idx - 1] = '\0';

    // Try to allocate memory and copy over message. If message buffer is
    // still in use, ignore the entire message.
    if (msg_flag == 0) {
      msg_ptr = (char *)pvPortMalloc(idx * sizeof(char));

      // If malloc returns 0 (out of memory), throw an error and reset
      configASSERT(msg_ptr);

      // Copy message
      memcpy(msg_ptr, buf, idx);
```

```
        // Notify other task that message is ready
        msg_flag = 1;
      }

      // Reset receive buffer and index counter
      memset(buf, 0, buf_len);
      idx = 0;
    }
  }
 }
}

// Task: print message whenever flag is set and free buffer
void printMessage(void *parameters) {
 while (1) {

   // Wait for flag to be set and print message
   if (msg_flag == 1) {
     Serial.println(msg_ptr);

     // Give amount of free heap memory (uncomment if you'd like to see it)
  Serial.print("Free heap (bytes): ");
   Serial.println(xPortGetFreeHeapSize());
     // Free buffer, set pointer to null, and clear flag
     vPortFree(msg_ptr);
```

```
      msg_ptr = NULL;

      msg_flag = 0;

    }

  }

}


//*****************************************************************************

// Main (runs as its own task with priority 1 on core 1)


void setup() {


  // Configure Serial

  Serial.begin(9600);


  // Wait a moment to start (so we don't miss Serial output)

  //vTaskDelay(1000 / portTICK_PERIOD_MS);

  Serial.println();

  Serial.println("---FreeRTOS Heap Demo---");

  Serial.println("Enter a string");


  // Start Serial receive task

  xTaskCreate(readSerial,"Read Serial",

                        1024,

                        NULL,

                        1,

                        NULL);
  // Start Serial print task
  xTaskCreate(printMessage,
                       "Print Message",
                       1024,
                       NULL,
                       1,
                       NULL);
  vTaskStartScheduler();
  // Delete "setup and loop" task
 // vTaskDelete(NULL);
}

void loop() {
  // Execution should never get here
}
```