Patrick Austin
CPE 301 – 1104
Assignment # 5
10/10/2016

Assignment description:

In this lab we worked through a number of problems from the textbook involving Arduino programming. We tinkered further with the Arduino built-in blink function and its ANSI C equivalent, we set up the Arduino serial I/O functions with putty, and then we used the functions to assist in debugging a series of sample programs from the textbook.

Problems encountered:

Other than getting used to using putty, which was a bit temperamental at the outset, problems were minor. As you'll probably see, I did get a bit confused about ambiguity in some of the questions, where we are to debug a blinking program to restore the blinking that was 'meant', but exactly what was meant is not clear from the code. I tried to give flexible answers that could accommodate a variety of interpretations of what the problem was trying to ask. In any case all the programs in question had logical errors which prevented any blinking at all, which I certainly fixed in my answers.

Lessons learned:

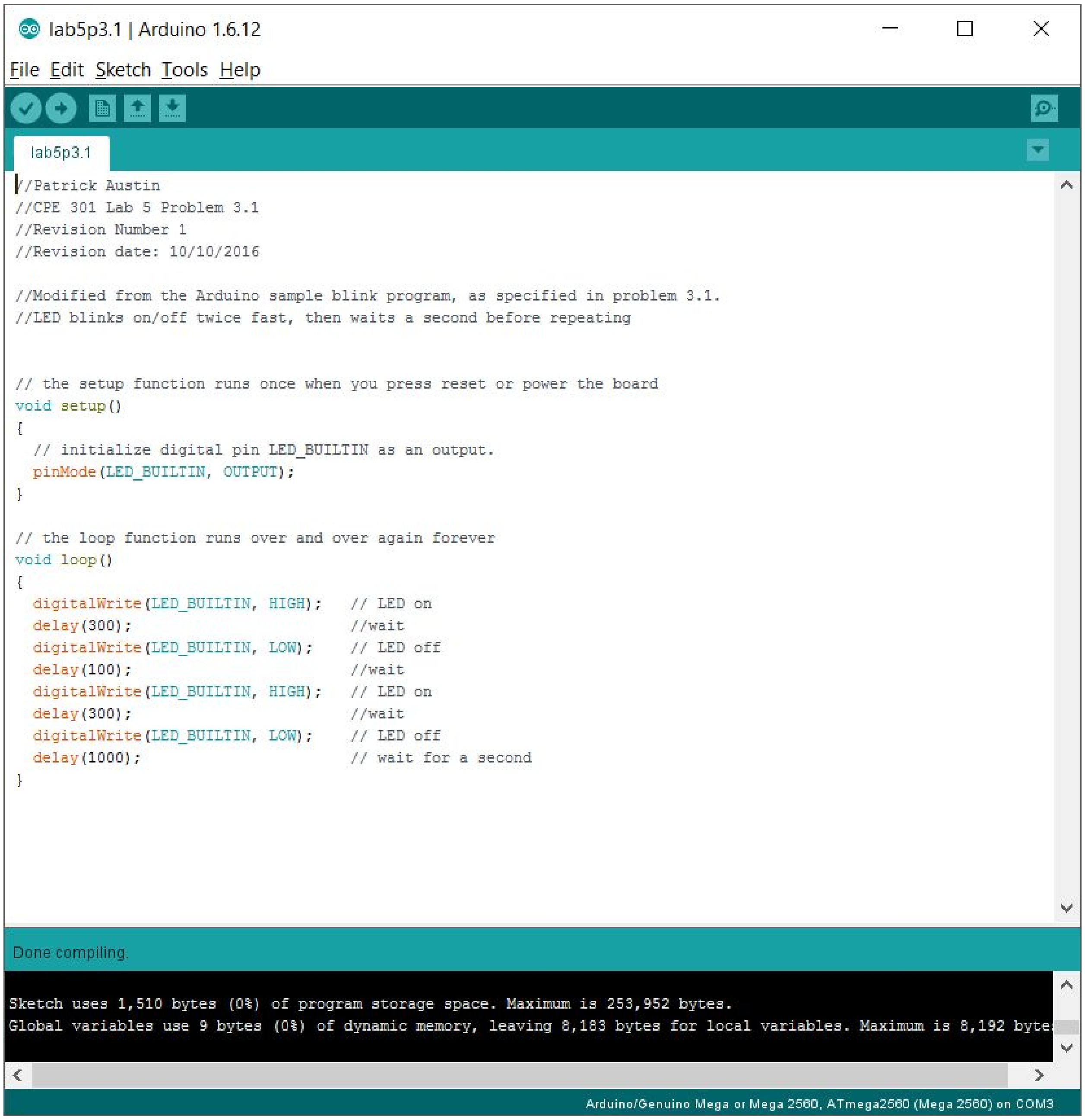
I got more review of basic manipulation of blink-type programs. Did initial setup and got some conceptual understanding for putty and serial I/O, which is a topic which we are just beginning to discuss in lecture and will surely be of continued relevance going forward. Using the serial functions to debug will be useful going forward, although I look forward to being able to implement serial functions without having to use the Arduino built-ins, as was directed in this lab.

Description of completed lab:

See the completed problems below. Manipulated Arduino-demo Blink program, textbook ANSI C blink program, and debugged a variety of erroneous blink programs. All told it's 20 pages of pure CPE 301 lab 5 hotness.

3.1. Here is the specified program:

```
//Patrick Austin
//CPE 301 Lab 5 Problem 3.1
//Revision Number 1
//Revision date: 10/10/2016

//Modified from the Arduino sample blink program, as specified in problem 3.1.
//LED blinks on/off twice fast, then waits a second before repeating


// the setup function runs once when you press reset or power the board
void setup()
{
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);   // LED on
  delay(300);                        //wait
  digitalWrite(LED_BUILTIN, LOW);   // LED off
  delay(100);                        //wait
  digitalWrite(LED_BUILTIN, HIGH);   // LED on
  delay(300);                        //wait
  digitalWrite(LED_BUILTIN, LOW);   // LED off
  delay(1000);                       // wait for a second
}
```



```
//Patrick Austin
//CPE 301 Lab 5 Problem 3.1
//Revision Number 1
//Revision date: 10/10/2016

//Modified from the Arduino sample blink program, as specified in problem 3.1.
//LED blinks on/off twice fast, then waits a second before repeating


// the setup function runs once when you press reset or power the board
void setup()
{
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);   // LED on
  delay(300);                        //wait
  digitalWrite(LED_BUILTIN, LOW);    // LED off
  delay(100);                        //wait
  digitalWrite(LED_BUILTIN, HIGH);   // LED on
  delay(300);                        //wait
  digitalWrite(LED_BUILTIN, LOW);    // LED off
  delay(1000);                       // wait for a second
}
```

Done compiling.

```
Sketch uses 1,510 bytes (0%) of program storage space. Maximum is 253,952 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 8,183 bytes for local variables. Maximum is 8,192 byte
```

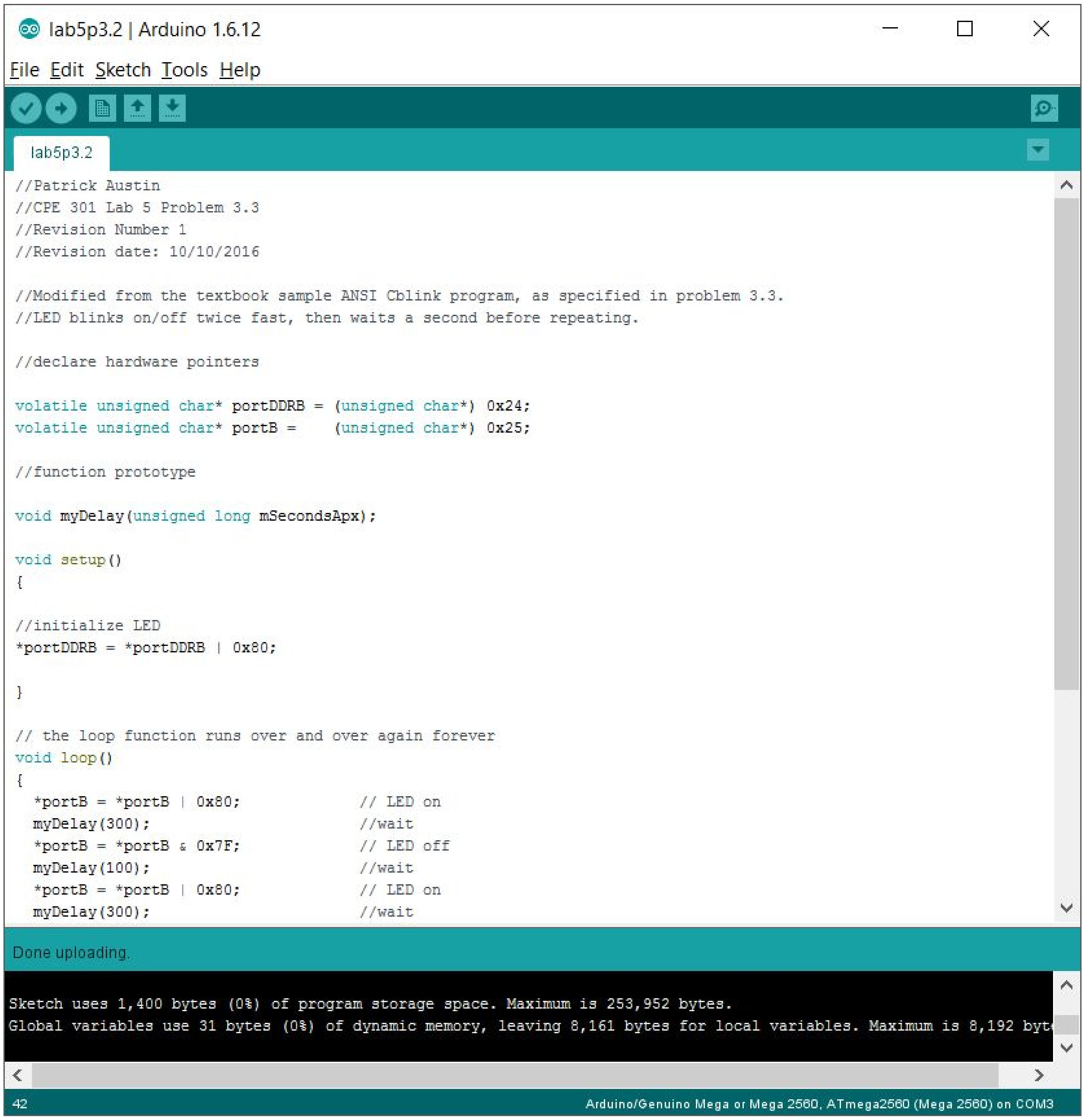
Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM3

3.2. As shown in the above screenshot, the program requires 1510 bytes of program storage space and 9 bytes of dynamic memory.

3.3. Here is the specified program:

```
//Patrick Austin
//CPE 301 Lab 5 Problem 3.3
//Revision Number 1
//Revision date: 10/10/2016

//Modified from the textbook sample ANSI Cblink program, as specified in problem 3.3.
//LED blinks on/off twice fast, then waits a second before repeating.

//declare hardware pointers

volatile unsigned char* portDDRB = (unsigned char*) 0x24;
volatile unsigned char* portB =         (unsigned char*) 0x25;

//function prototype

void myDelay(unsigned long mSecondsApx);

void setup()
{

//initialize LED
*portDDRB = *portDDRB | 0x80;

}

// the loop function runs over and over again forever
void loop()
{
  *portB = *portB | 0x80;         // LED on
  myDelay(300);                   //wait
  *portB = *portB & 0x7F;                 // LED off
  myDelay(100);                   //wait
  *portB = *portB | 0x80;         // LED on
  myDelay(300);                   //wait
  *portB = *portB & 0x7F;                 // LED off
  myDelay(1000);                  // wait for a second
}
```

```
void myDelay (unsigned long mSecondsApx)
{

volatile unsigned long i;
unsigned long endTime = 1000 * mSecondsApx;

for (  i = 0; i < endTime; i++ );

}
```



```
//Patrick Austin
//CPE 301 Lab 5 Problem 3.3
//Revision Number 1
//Revision date: 10/10/2016

//Modified from the textbook sample ANSI Cblink program, as specified in problem 3.3.
//LED blinks on/off twice fast, then waits a second before repeating.

//declare hardware pointers

volatile unsigned char* portDDRB = (unsigned char*) 0x24;
volatile unsigned char* portB =    (unsigned char*) 0x25;

//function prototype

void myDelay(unsigned long mSecondsApx);

void setup()
{

//initialize LED
*portDDRB = *portDDRB | 0x80;

}

// the loop function runs over and over again forever
void loop()
{
  *portB = *portB | 0x80;             // LED on
  myDelay(300);                       //wait
  *portB = *portB & 0x7F;             // LED off
  myDelay(100);                       //wait
  *portB = *portB | 0x80;             // LED on
  myDelay(300);                       //wait
```

Done uploading.

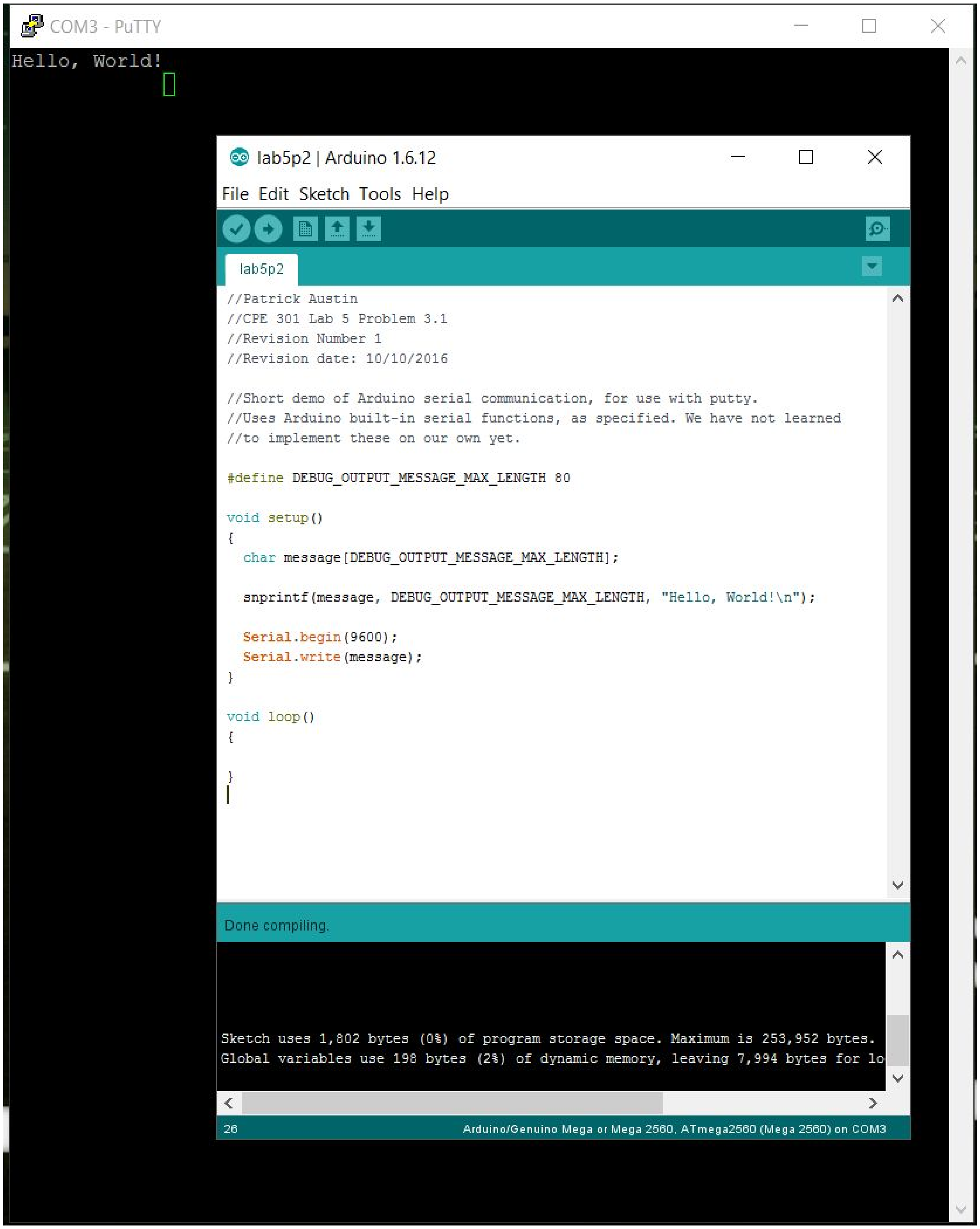
```
Sketch uses 1,400 bytes (0%) of program storage space. Maximum is 253,952 bytes.
Global variables use 31 bytes (0%) of dynamic memory, leaving 8,161 bytes for local variables. Maximum is 8,192 byt
```

42
Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM3

3.4. As shown in the above screenshot, the program requires 1400 bytes of program storage space and 31 bytes of dynamic memory.

Part 2

```
//Patrick Austin
//CPE 301 Lab 5 Problem 3.1
//Revision Number 1
//Revision date: 10/10/2016

//Short demo of Arduino serial communication, for use with putty.
//Uses Arduino built-in serial functions, as specified. We have not learned
//to implement these on our own yet. Adapted from textbook code.

#define DEBUG_OUTPUT_MESSAGE_MAX_LENGTH 80

void setup()
{
  char message[DEBUG_OUTPUT_MESSAGE_MAX_LENGTH];

  snprintf(message, DEBUG_OUTPUT_MESSAGE_MAX_LENGTH, "Hello, World!\n");

  Serial.begin(9600);
  Serial.write(message);
}

void loop()
{

}
```


COM3 - PuTTY
lab5p2 | Arduino 1.6.12
File Edit Sketch Tools Help
lab5p2
Done compiling.
26
Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM3

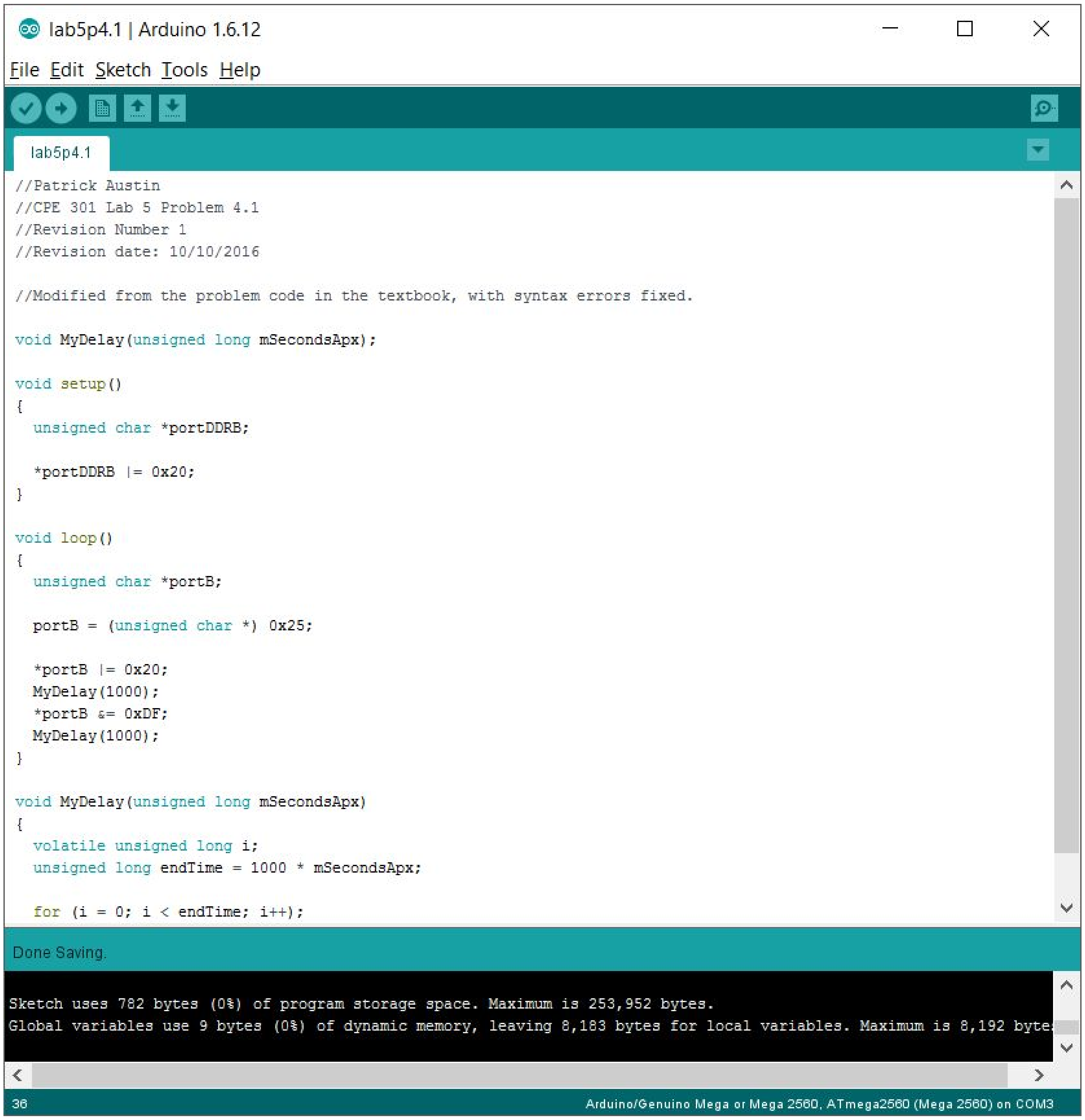
```
Hello, World!
```

```
//Patrick Austin
//CPE 301 Lab 5 Problem 3.1
//Revision Number 1
//Revision date: 10/10/2016

//Short demo of Arduino serial communication, for use with putty.
//Uses Arduino built-in serial functions, as specified. We have not learned
//to implement these on our own yet.

#define DEBUG_OUTPUT_MESSAGE_MAX_LENGTH 80

void setup()
{
  char message[DEBUG_OUTPUT_MESSAGE_MAX_LENGTH];

  snprintf(message, DEBUG_OUTPUT_MESSAGE_MAX_LENGTH, "Hello, World!\n");

  Serial.begin(9600);
  Serial.write(message);
}

void loop()
{

}
```

```
Sketch uses 1,802 bytes (0%) of program storage space. Maximum is 253,952 bytes.
Global variables use 198 bytes (2%) of dynamic memory, leaving 7,994 bytes for lo
```

4.1. Here is the corrected code:

```
//Patrick Austin
//CPE 301 Lab 5 Problem 4.1
//Revision Number 1
//Revision date: 10/10/2016

//Modified from the problem code in the textbook, with syntax errors fixed.

void MyDelay(unsigned long mSecondsApx);

void setup()
{
  unsigned char *portDDRB;

  *portDDRB |= 0x20;
}

void loop()
{
  unsigned char *portB;

  portB = (unsigned char *) 0x25;

  *portB |= 0x20;
  MyDelay(1000);
  *portB &= 0xDF;
  MyDelay(1000);
}

void MyDelay(unsigned long mSecondsApx)
{
  volatile unsigned long i;
  unsigned long endTime = 1000 * mSecondsApx;

  for (i = 0; i < endTime; i++);
}

//There was a syntax error at each of the myDelay calls, using the wrong type of brackets for a
//function call.
```



```
//Patrick Austin
//CPE 301 Lab 5 Problem 4.1
//Revision Number 1
//Revision date: 10/10/2016

//Modified from the problem code in the textbook, with syntax errors fixed.

void MyDelay(unsigned long mSecondsApx);

void setup()
{
  unsigned char *portDDRB;

  *portDDRB |= 0x20;
}

void loop()
{
  unsigned char *portB;

  portB = (unsigned char *) 0x25;

  *portB |= 0x20;
  MyDelay(1000);
  *portB &= 0xDF;
  MyDelay(1000);
}

void MyDelay(unsigned long mSecondsApx)
{
  volatile unsigned long i;
  unsigned long endTime = 1000 * mSecondsApx;

  for (i = 0; i < endTime; i++);
```

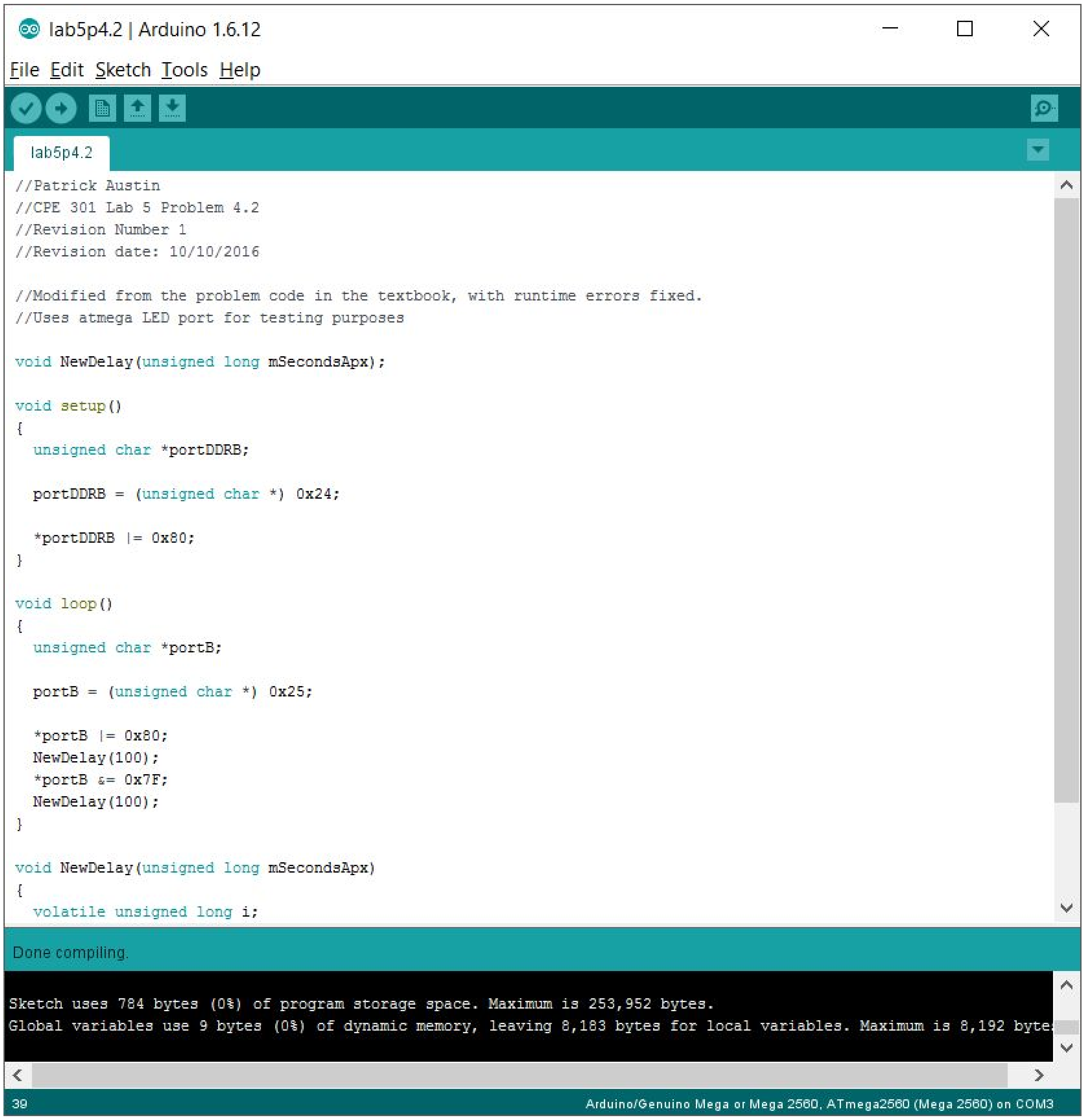
Done Saving.

```
Sketch uses 782 bytes (0%) of program storage space. Maximum is 253,952 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 8,183 bytes for local variables. Maximum is 8,192 byte
```

36 Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM3

4.2. Here is the corrected code (modified to use the atmega LED port):

```
//Patrick Austin
//CPE 301 Lab 5 Problem 4.2
//Revision Number 1
//Revision date: 10/10/2016

//Modified from the problem code in the textbook, with runtime errors fixed.
//Uses atmega LED port for testing purposes

void NewDelay(unsigned long mSecondsApx);

void setup()
{
  unsigned char *portDDRB;

  portDDRB = (unsigned char *) 0x24;

  *portDDRB |= 0x80;
}

void loop()
{
  unsigned char *portB;

  portB = (unsigned char *) 0x25;

  *portB |= 0x80;
  NewDelay(100);
  *portB &= 0x7F;
  NewDelay(100);
}

void NewDelay(unsigned long mSecondsApx)
{
  volatile unsigned long i;
  unsigned long endTime = 1000 * mSecondsApx;

  for (i = 0; i < endTime; i++);
}
```



```
//Patrick Austin
//CPE 301 Lab 5 Problem 4.2
//Revision Number 1
//Revision date: 10/10/2016

//Modified from the problem code in the textbook, with runtime errors fixed.
//Uses atmega LED port for testing purposes

void NewDelay(unsigned long mSecondsApx);

void setup()
{
  unsigned char *portDDRB;

  portDDRB = (unsigned char *) 0x24;

  *portDDRB |= 0x80;
}

void loop()
{
  unsigned char *portB;

  portB = (unsigned char *) 0x25;

  *portB |= 0x80;
  NewDelay(100);
  *portB &= 0x7F;
  NewDelay(100);
}

void NewDelay(unsigned long mSecondsApx)
{
  volatile unsigned long i;
```

Done compiling.

```
Sketch uses 784 bytes (0%) of program storage space. Maximum is 253,952 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 8,183 bytes for local variables. Maximum is 8,192 byte
```

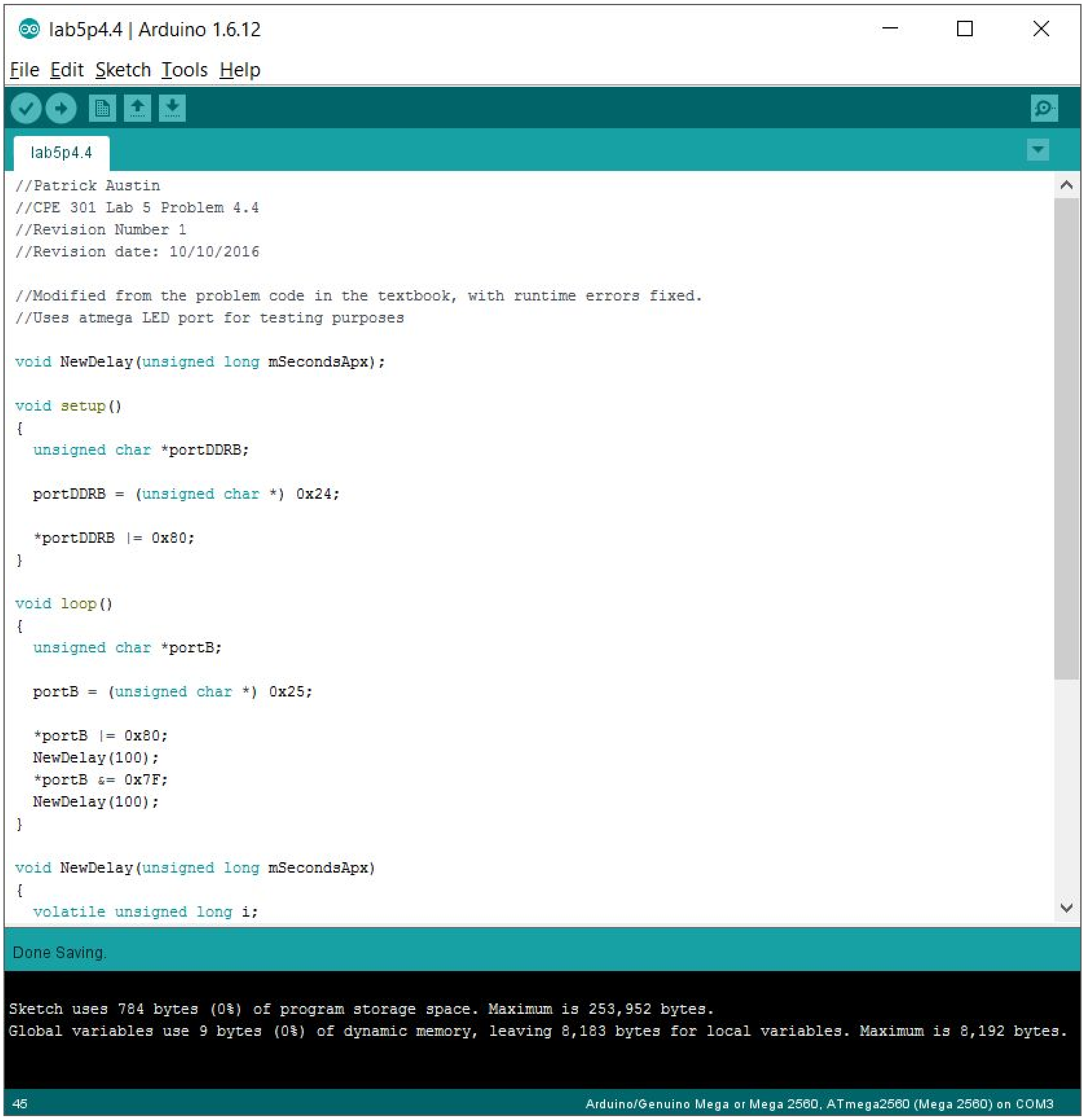
39
Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM3

4.3. The original program used chars instead of longs for the mSecondsApx and i variables. This caused the logic of the for loop to never complete. Using longs instead restored normal program operation.

4.4. Here is the corrected code (modified to use the atmega LED port):

```
//Patrick Austin
//CPE 301 Lab 5 Problem 4.4
//Revision Number 1
//Revision date: 10/10/2016

//Modified from the problem code in the textbook, with runtime errors fixed.
//Uses atmega LED port for testing purposes

void NewDelay(unsigned long mSecondsApx);

void setup()
{
  unsigned char *portDDRB;

  portDDRB = (unsigned char *) 0x24;

  *portDDRB |= 0x80;
}

void loop()
{
  unsigned char *portB;

  portB = (unsigned char *) 0x25;

  *portB |= 0x80;
  NewDelay(100);
  *portB &= 0x7F;
  NewDelay(100);
}
```

```
void NewDelay(unsigned long mSecondsApx)
{
  volatile unsigned long i;
  unsigned long j;
  unsigned long k;
  unsigned long endTime = 100 * mSecondsApx;

  for (i = 0; i < endTime; i++)
  {
          j = 10;
          do
          {
          j = j - 1;
          k = i / j;
          } while (k > 0);
  }
}
```



```
//Patrick Austin
//CPE 301 Lab 5 Problem 4.4
//Revision Number 1
//Revision date: 10/10/2016

//Modified from the problem code in the textbook, with runtime errors fixed.
//Uses atmega LED port for testing purposes

void NewDelay(unsigned long mSecondsApx);

void setup()
{
  unsigned char *portDDRB;

  portDDRB = (unsigned char *) 0x24;

  *portDDRB |= 0x80;
}

void loop()
{
  unsigned char *portB;

  portB = (unsigned char *) 0x25;

  *portB |= 0x80;
  NewDelay(100);
  *portB &= 0x7F;
  NewDelay(100);
}

void NewDelay(unsigned long mSecondsApx)
{
  volatile unsigned long i;
```

Done Saving.

```
Sketch uses 784 bytes (0%) of program storage space. Maximum is 253,952 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 8,183 bytes for local variables. Maximum is 8,192 bytes.
```

45 Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM3

4.5. The minimum needed to restore blinking function at all is to make variable j a long instead of a char. As a char, the logic of the do/while loop never completes. The change to a long restores (very slow) blinking.

The prompt seems ambiguous to me: in absence of comments or any other specifics, I'm not sure how to know how the LED was "meant" to blink, other than that it should blink. If one is supposed to take issue with the slowness of the blinking, removing the divide operation extremely increases the speed of blinking. I didn't include this change in the solution above, but it would entail eg changing line

'k = i / j;'   to
'k = j;'

or somesuch.

If one is supposed to desire blinking in accord with the name of the mSecondsApx variable, where approx 1 ms * msSecondsApx should be delayed when the method is called, then one can change line

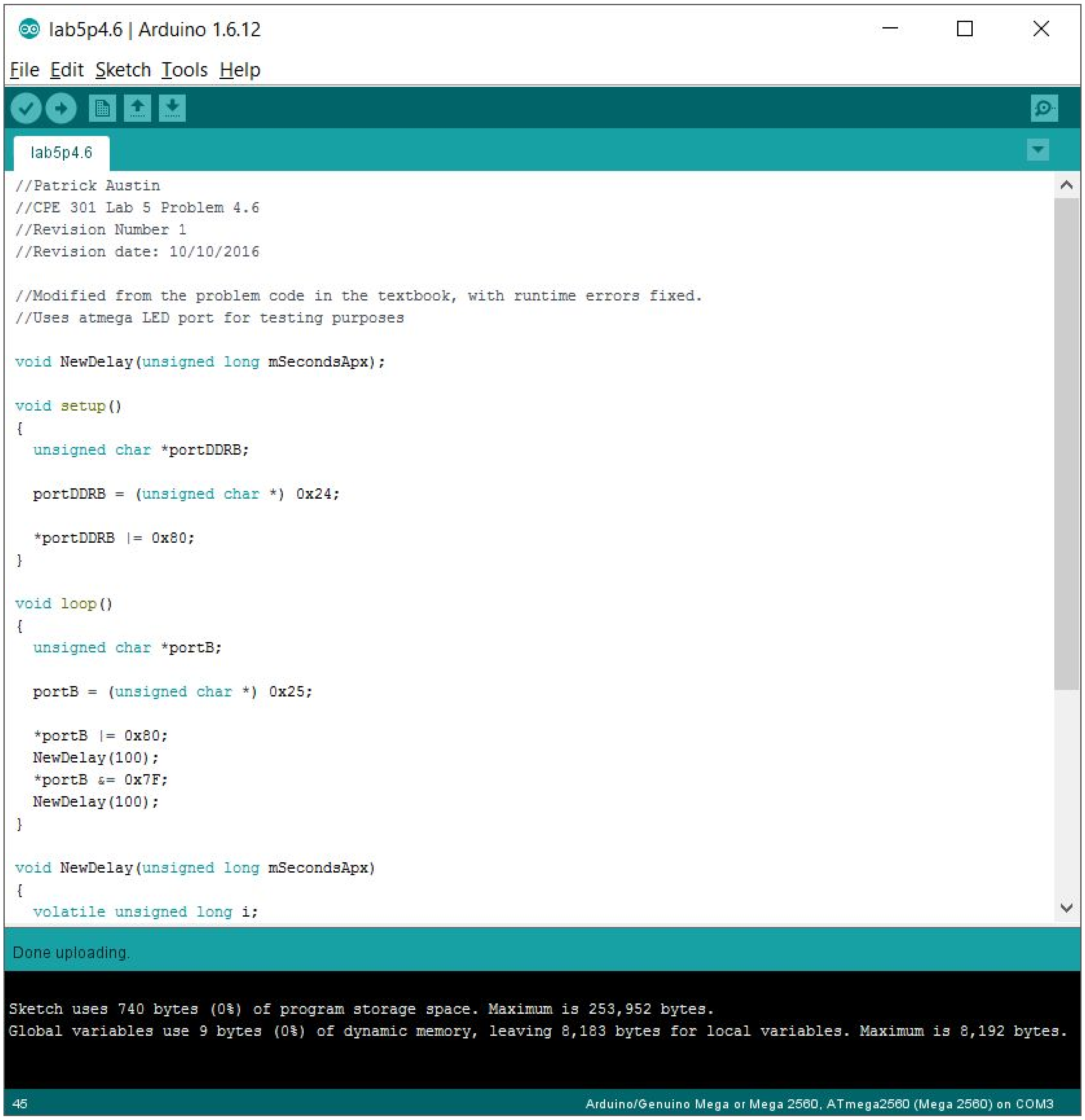
'unsigned long endTime = 100 * mSecondsApx;'   to
'unsigned long endTime = 1000 * mSecondsApx;'

, remove the contents of the for loop, and remove the unused variables, which restores the normal delay function operation. Again, this question and the next in the same vein seem ambiguous to me, so I hope you forgive my confusion.

4.6: Here is the corrected code (modified to use the atmega LED port):

```
//Patrick Austin
//CPE 301 Lab 5 Problem 4.6
//Revision Number 1
//Revision date: 10/10/2016

//Modified from the problem code in the textbook, with runtime errors fixed.
//Uses atmega LED port for testing purposes

void NewDelay(unsigned long mSecondsApx);

void setup()
{
  unsigned char *portDDRB;

  portDDRB = (unsigned char *) 0x24;

  *portDDRB |= 0x80;
}

void loop()
{
  unsigned char *portB;

  portB = (unsigned char *) 0x25;

  *portB |= 0x80;
  NewDelay(100);
  *portB &= 0x7F;
  NewDelay(100);
}
```

```
void NewDelay(unsigned long mSecondsApx)
{
  volatile unsigned long i;
  unsigned char j = 0;
  unsigned long endTime = 100 * mSecondsApx;

  i = 0;
  while ( j == 0 )
  {
          i++;
          if ( i == endTime)
          {
          j = 1;
          }
  }
}
```



```
//Patrick Austin
//CPE 301 Lab 5 Problem 4.6
//Revision Number 1
//Revision date: 10/10/2016

//Modified from the problem code in the textbook, with runtime errors fixed.
//Uses atmega LED port for testing purposes

void NewDelay(unsigned long mSecondsApx);

void setup()
{
  unsigned char *portDDRB;

  portDDRB = (unsigned char *) 0x24;

  *portDDRB  |= 0x80;
}

void loop()
{
  unsigned char *portB;

  portB = (unsigned char *) 0x25;

  *portB  |= 0x80;
  NewDelay(100);
  *portB &= 0x7F;
  NewDelay(100);
}

void NewDelay(unsigned long mSecondsApx)
{
  volatile unsigned long i;
```

Done uploading.

```
Sketch uses 740 bytes (0%) of program storage space. Maximum is 253,952 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 8,183 bytes for local variables. Maximum is 8,192 bytes.
```

45 Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM3

4.7. The minimum needed to restore blinking to the program was to change the = operators in the while and if statements to == operators, so that comparison was actually taking place and the loop would terminate. In this case it is not necessary to make j a long, since it is only has two possible values which are being used for comparison.

As in 4.5, I think this question is somewhat ambiguous. This change alone results in very fast blinking. Is that what was “meant” to happen? I’m not sure. If one was supposed to desire slower blinking, such that NewDelay caused a delay of about 1ms * mSecondsApx, one could make the NewDelay function look more like the template provided in the textbook, or multiply mSecondsApx by a larger value so that more iterations of the while loop take place.

Again, forgive my confusion.