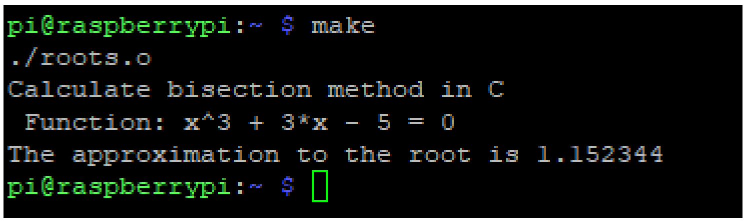
prac_2_lphtum003.pdf

Part 1

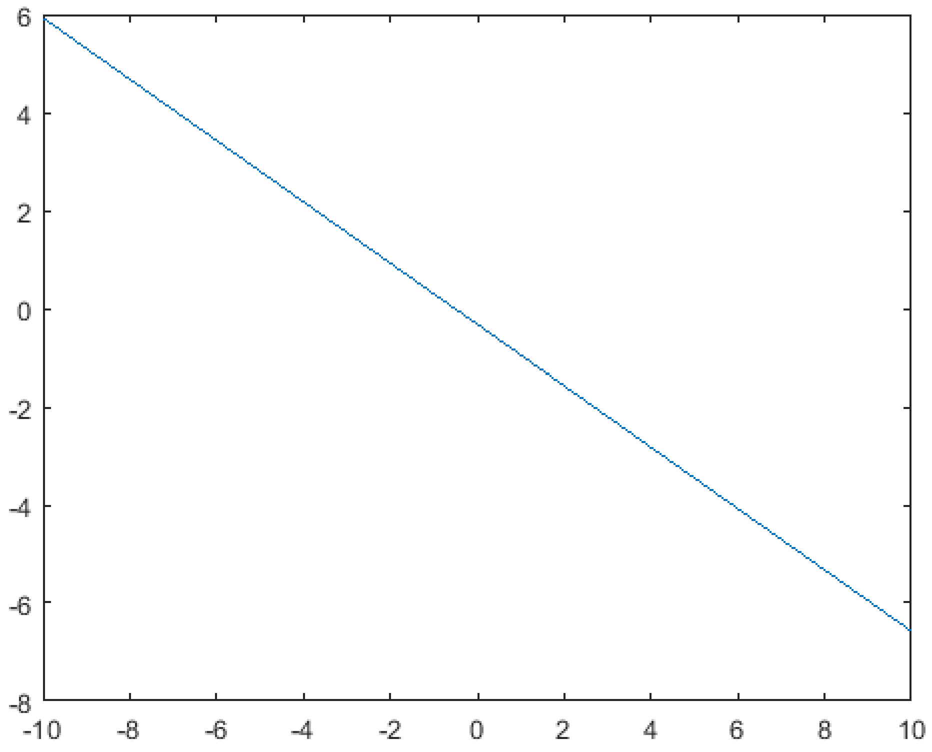


```
pi@raspberrypi:~ $ make
./roots.o
Calculate bisection method in C
 Function: x^3 + 3*x - 5 = 0
The approximation to the root is 1.152344
pi@raspberrypi:~ $ 
```

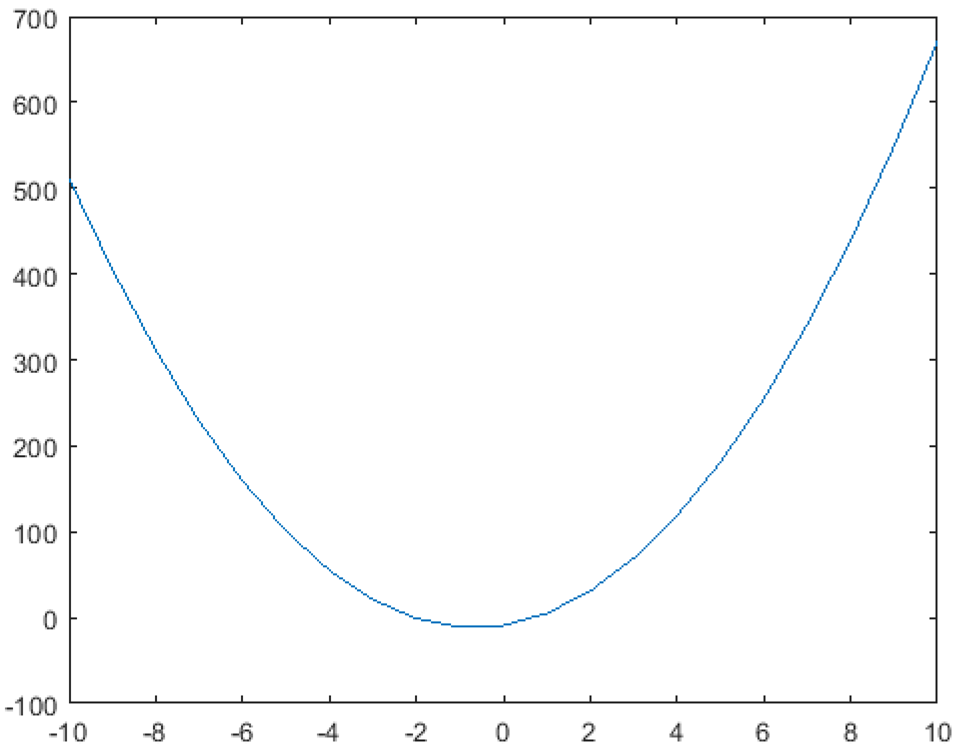
Part 2

The algorithm is accurate depending on the number of iterations selected too many and it gives you naught and too little then you get a midpoint of the bisection window.

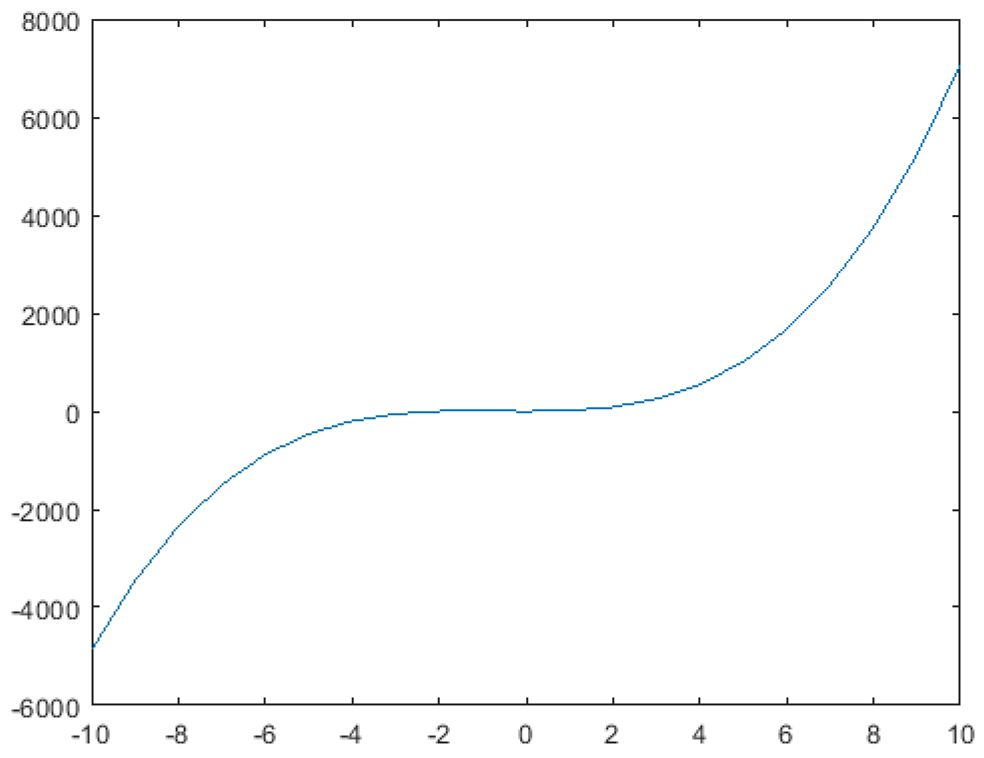
```
Trial>> x = [-0.6254 -0.3234]

x =

   -0.6254   -0.3234

Trial>> roots(x)

ans =

   -0.5171
```



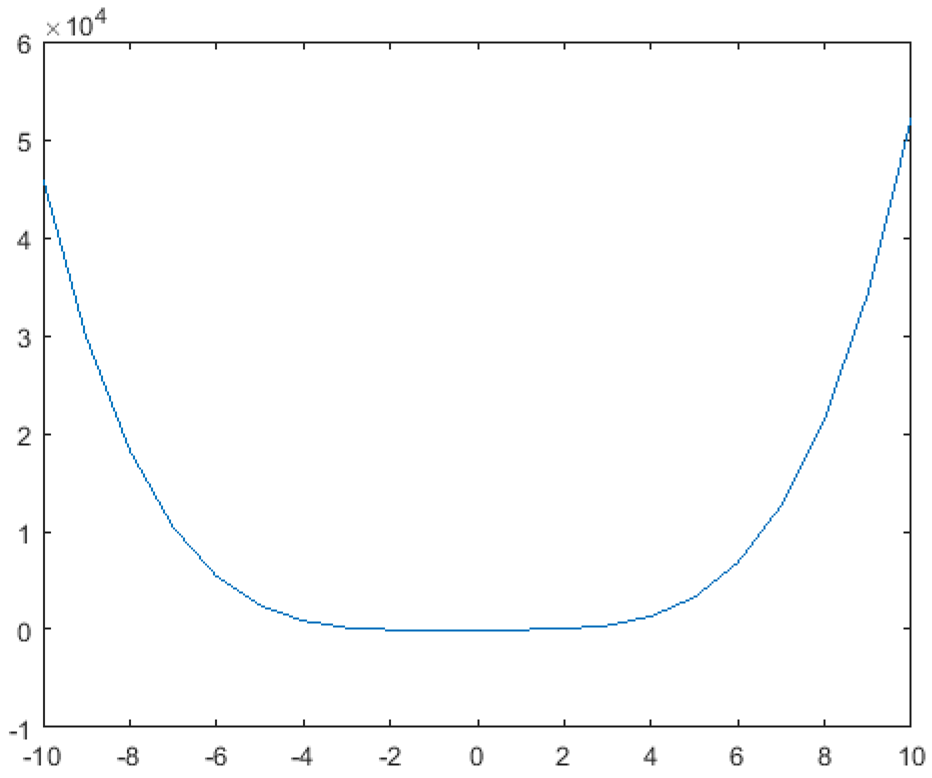
```
Trial>> x = [6 8 -9]

x =

     6     8    -9

Trial>> roots(x)

ans =

   -2.0611
    0.7278
```



```
Trial>> x = [6 11 -3 -2]

x =

     6    11    -3    -2

Trial>> roots(x)

ans =

   -2.0000
    0.5000
   -0.3333
```



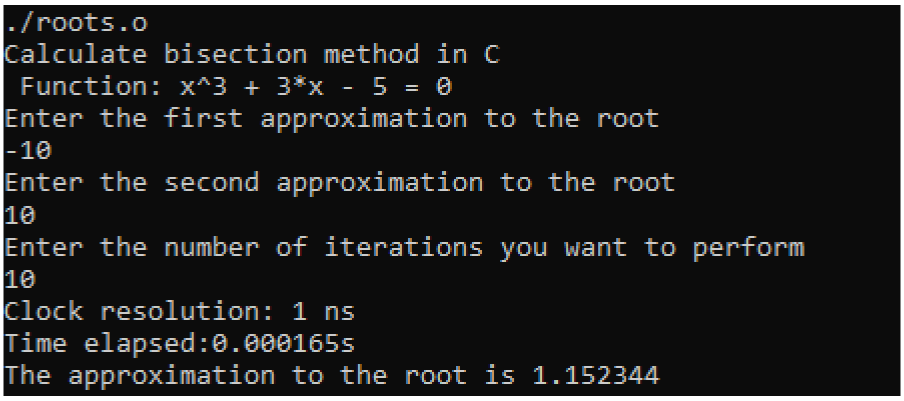
```
Trial>> x = [5 3 -9 10 -25]

x =

     5     3    -9    10   -25

Trial>> roots(x)

ans =

  -2.2430 + 0.0000i
   1.4205 + 0.0000i
   0.1113 + 1.2478i
   0.1113 - 1.2478i
```



Part 3

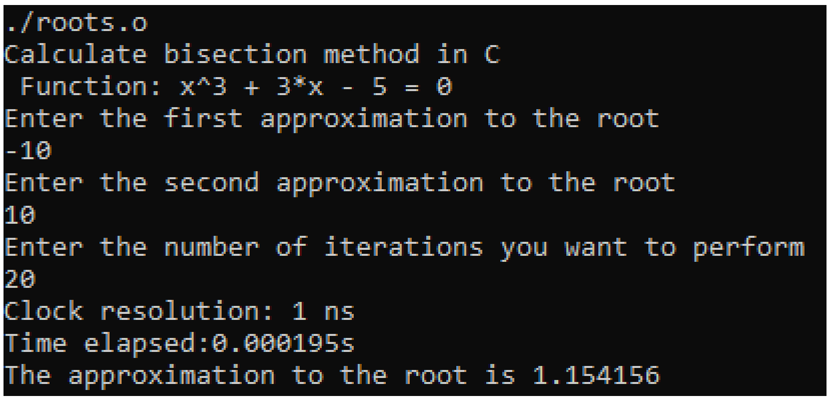
To test the performance I kept the function and end points constant and only changed the number of iterations. In doing so I found that when the iterations were increased the answer ended up being wrong and that the amount of time taken also increased. However my desktop proved to be faster in every increment of the number of iterations.

```
Calculate bisection method in C
 Function: x^3 + 3*x - 5 = 0
Enter the first approximation to the root
-10
Enter the second approximation to the root
10
Enter the number of iterations you want to perform
10
Clock resolution: 394.739 ns
Time elapsed:0.000004s
The approximation to the root is 1.152344
```

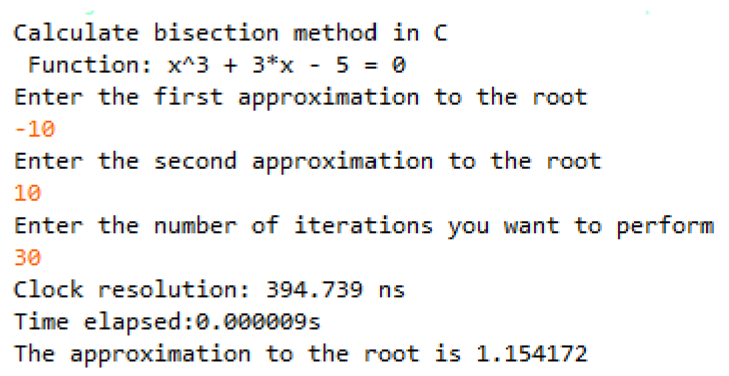


```
./roots.o
Calculate bisection method in C
 Function: x^3 + 3*x - 5 = 0
Enter the first approximation to the root
-10
Enter the second approximation to the root
10
Enter the number of iterations you want to perform
10
Clock resolution: 1 ns
Time elapsed:0.000165s
The approximation to the root is 1.152344
```

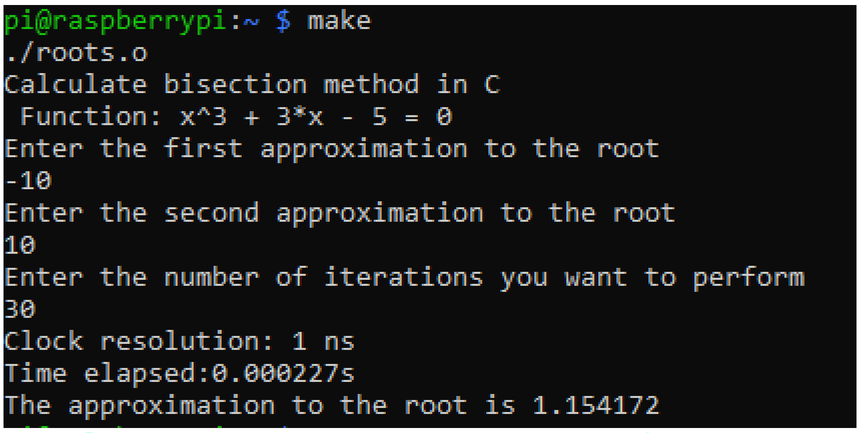
```
Calculate bisection method in C
 Function: x^3 + 3*x - 5 = 0
Enter the first approximation to the root
-10
Enter the second approximation to the root
10
Enter the number of iterations you want to perform
20
Clock resolution: 394.739 ns
Time elapsed:0.000007s
The approximation to the root is 1.154156
```



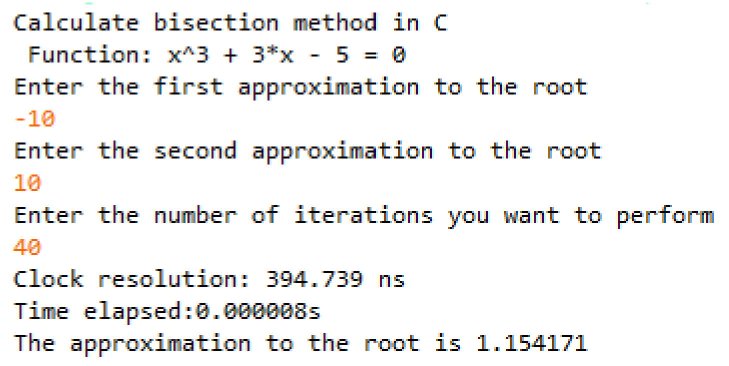
```
./roots.o
Calculate bisection method in C
 Function: x^3 + 3*x - 5 = 0
Enter the first approximation to the root
-10
Enter the second approximation to the root
10
Enter the number of iterations you want to perform
20
Clock resolution: 1 ns
Time elapsed:0.000195s
The approximation to the root is 1.154156
```



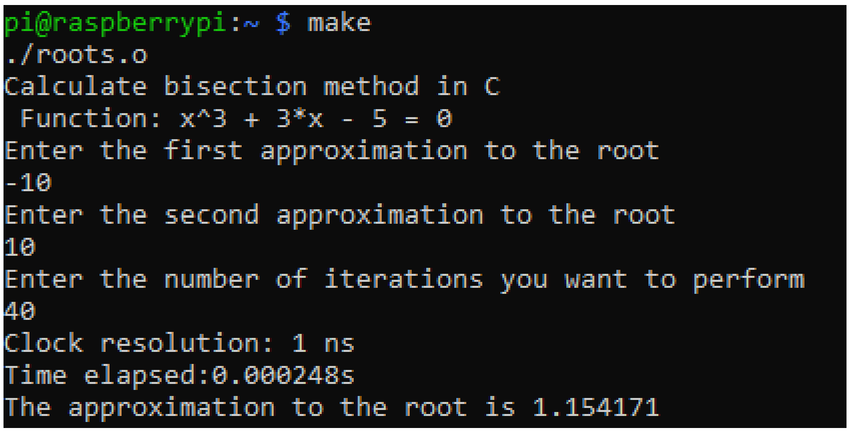
```
Calculate bisection method in C
 Function: x^3 + 3*x - 5 = 0
Enter the first approximation to the root
-10
Enter the second approximation to the root
10
Enter the number of iterations you want to perform
30
Clock resolution: 394.739 ns
Time elapsed:0.000009s
The approximation to the root is 1.154172
```



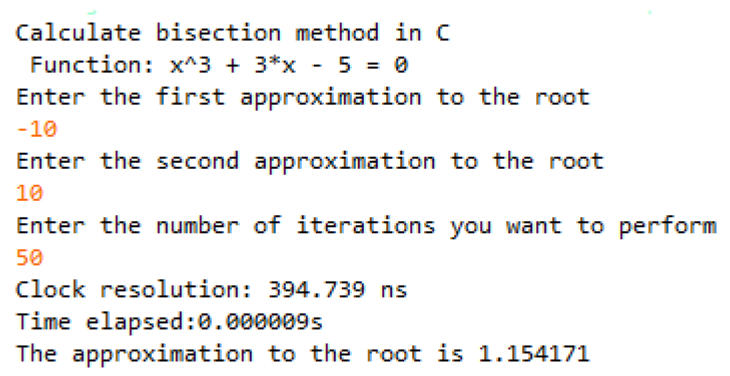
```
pi@raspberrypi:~ $ make
./roots.o
Calculate bisection method in C
 Function: x^3 + 3*x - 5 = 0
Enter the first approximation to the root
-10
Enter the second approximation to the root
10
Enter the number of iterations you want to perform
30
Clock resolution: 1 ns
Time elapsed:0.000227s
The approximation to the root is 1.154172
```



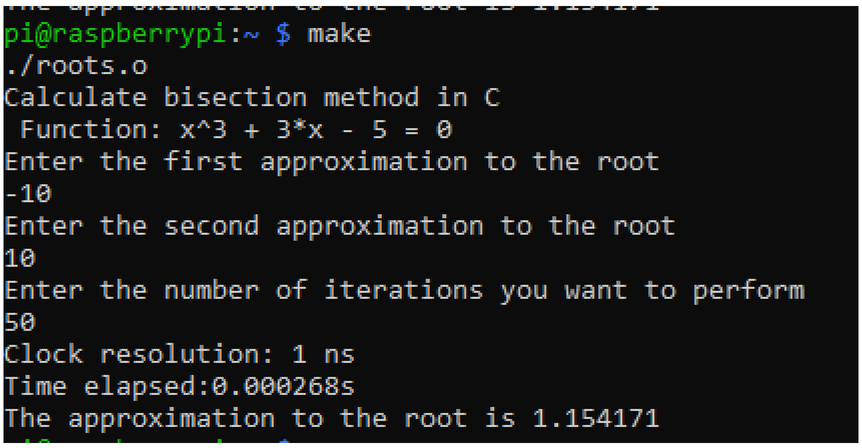
```
Calculate bisection method in C
 Function: x^3 + 3*x - 5 = 0
Enter the first approximation to the root
-10
Enter the second approximation to the root
10
Enter the number of iterations you want to perform
40
Clock resolution: 394.739 ns
Time elapsed:0.000008s
The approximation to the root is 1.154171
```



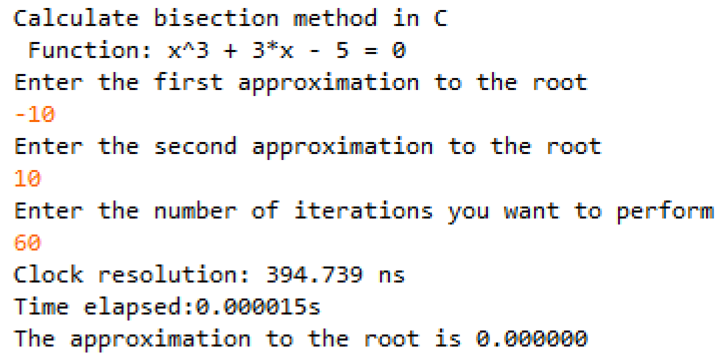
```
pi@raspberrypi:~ $ make
./roots.o
Calculate bisection method in C
 Function: x^3 + 3*x - 5 = 0
Enter the first approximation to the root
-10
Enter the second approximation to the root
10
Enter the number of iterations you want to perform
40
Clock resolution: 1 ns
Time elapsed:0.000248s
The approximation to the root is 1.154171
```



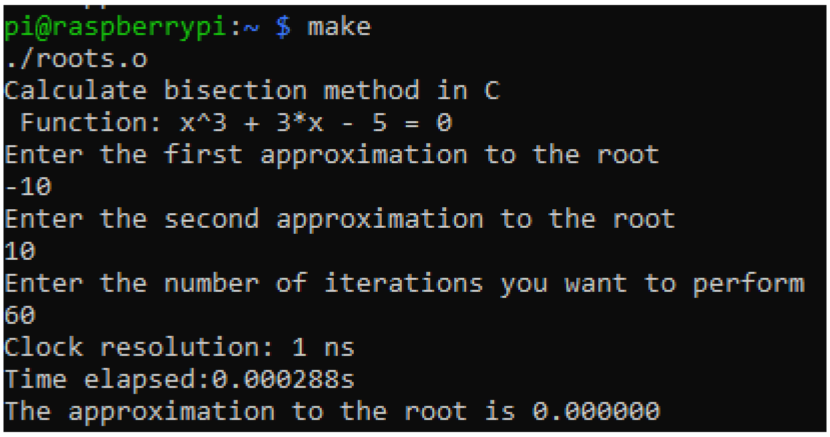
```
Calculate bisection method in C
 Function: x^3 + 3*x - 5 = 0
Enter the first approximation to the root
-10
Enter the second approximation to the root
10
Enter the number of iterations you want to perform
50
Clock resolution: 394.739 ns
Time elapsed:0.000009s
The approximation to the root is 1.154171
```



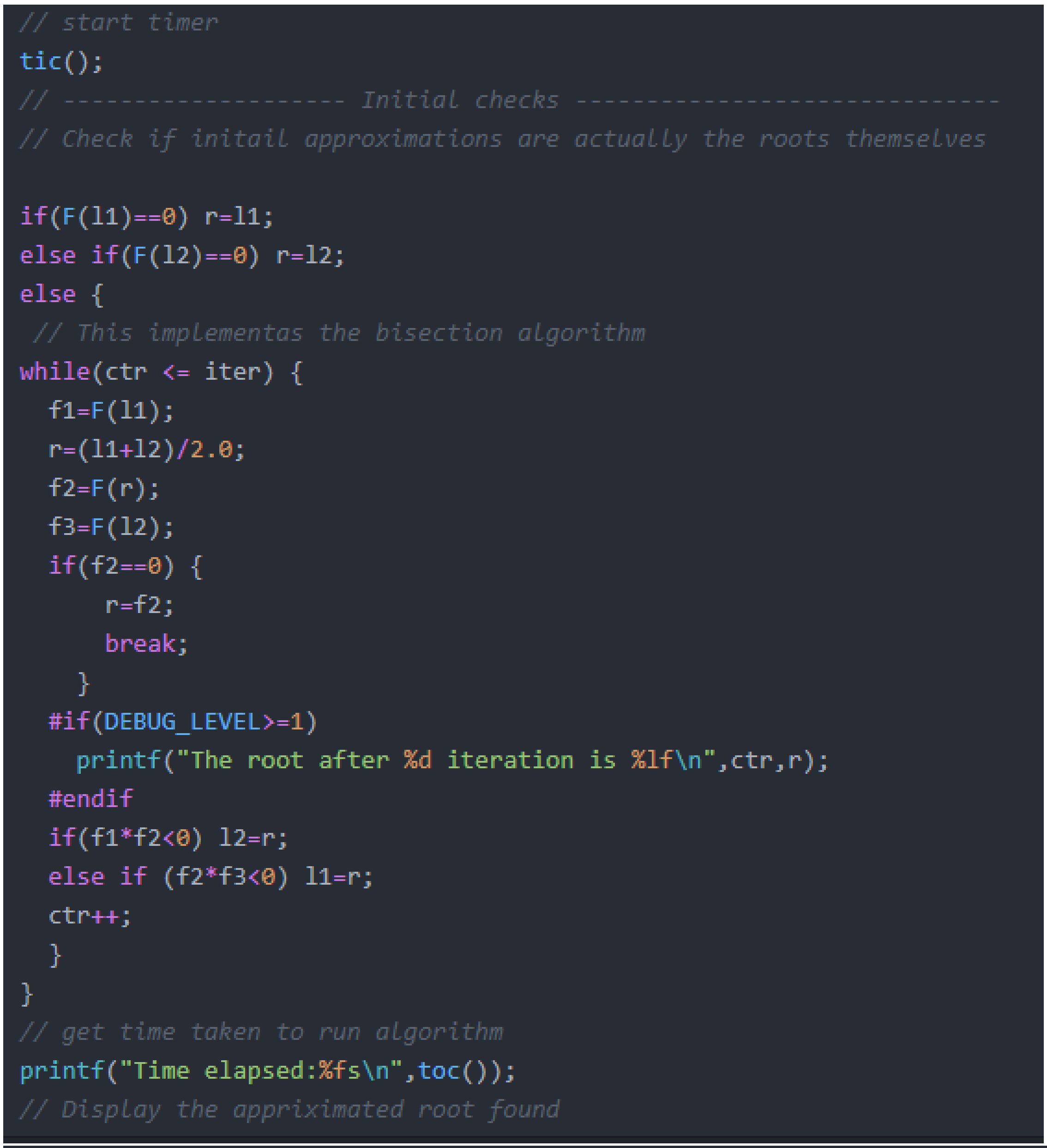
```
pi@raspberrypi:~ $ make
./roots.o
Calculate bisection method in C
 Function: x^3 + 3*x - 5 = 0
Enter the first approximation to the root
-10
Enter the second approximation to the root
10
Enter the number of iterations you want to perform
50
Clock resolution: 1 ns
Time elapsed:0.000268s
The approximation to the root is 1.154171
```



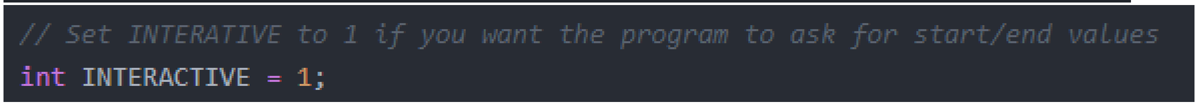
```
Calculate bisection method in C
 Function: x^3 + 3*x - 5 = 0
Enter the first approximation to the root
-10
Enter the second approximation to the root
10
Enter the number of iterations you want to perform
60
Clock resolution: 394.739 ns
Time elapsed:0.000015s
The approximation to the root is 0.000000
```



```
pi@raspberrypi:~ $ make
./roots.o
Calculate bisection method in C
 Function: x^3 + 3*x - 5 = 0
Enter the first approximation to the root
-10
Enter the second approximation to the root
10
Enter the number of iterations you want to perform
60
Clock resolution: 1 ns
Time elapsed:0.000288s
The approximation to the root is 0.000000
```



```c
// start timer
tic();
// -------------------- Initial checks ------------------------------
// Check if initail approximations are actually the roots themselves

if(F(l1)==0) r=l1;
else if(F(l2)==0) r=l2;
else {
 // This implementas the bisection algorithm
while(ctr <= iter) {
  f1=F(l1);
  r=(l1+l2)/2.0;
  f2=F(r);
  f3=F(l2);
  if(f2==0) {
      r=f2;
      break;
    }
  #if(DEBUG_LEVEL>=1)
    printf("The root after %d iteration is %lf\n",ctr,r);
  #endif
  if(f1*f2<0) l2=r;
  else if (f2*f3<0) l1=r;
  ctr++;
  }
}
// get time taken to run algorithm
printf("Time elapsed:%fs\n",toc());
// Display the apprixmated root found
```



```c
// Set INTERATIVE to 1 if you want the program to ask for start/end values
int INTERACTIVE = 1;
```