Parallel Programming

Laboratory 1

~ 2022 ~

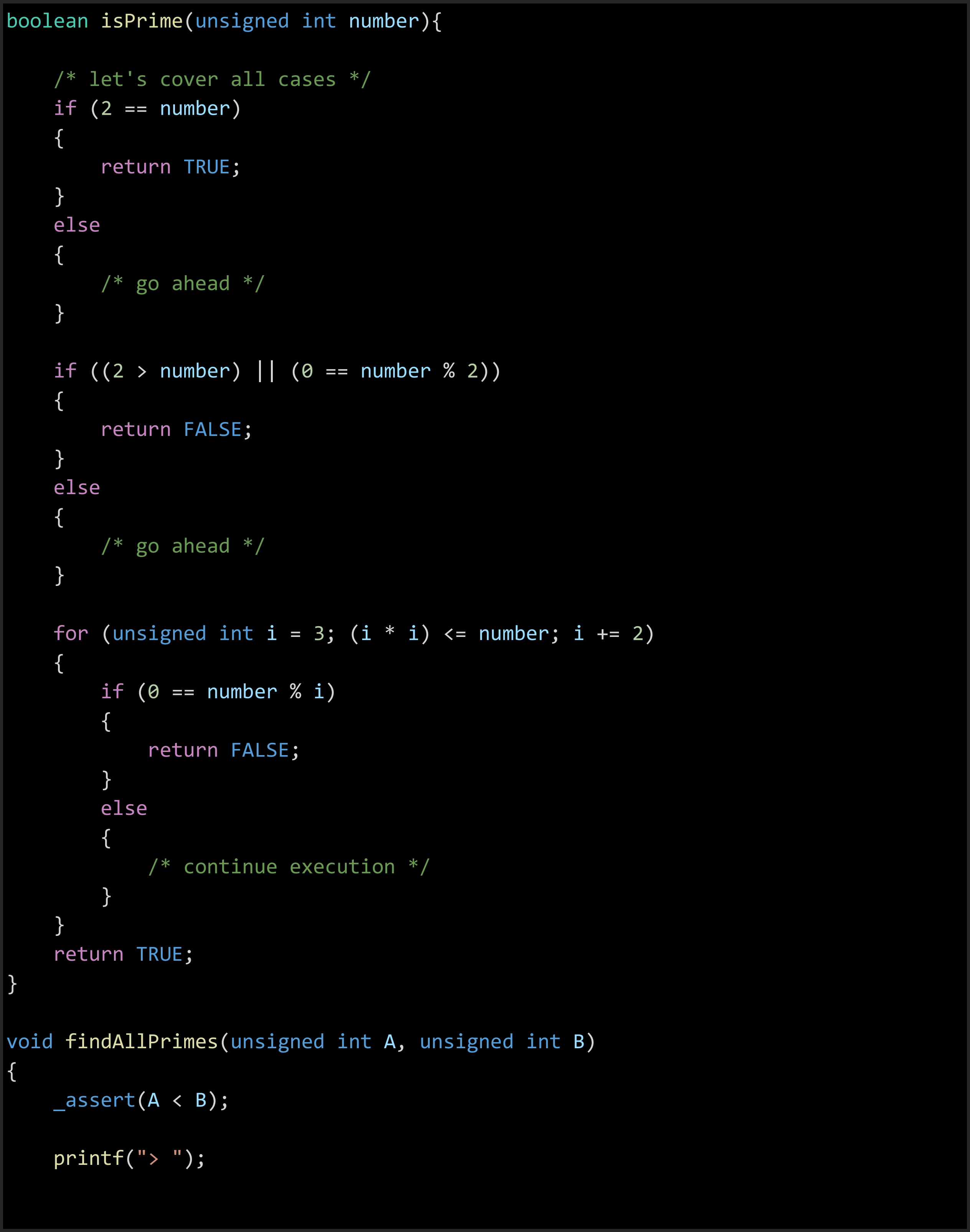
Balázs Benedek

Group 30444/1

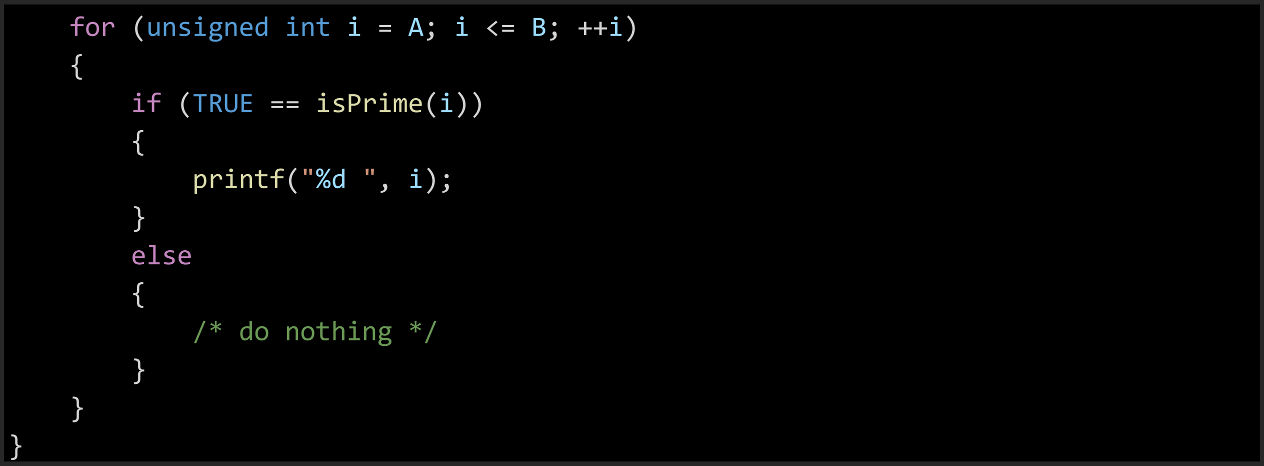
Problem 1

```
/***********************************************
*         Benedek Balazs - Group 30444
*                   ~ 2022 ~
************************************************
*
*       Technical University of Cluj-Napoca
*    Faculty of Automation and Computer Science
*
*              Parallel Programming
*
*     _     _            _        ____   ____
*    | |__ | | ___  ___ |_| ___  /    \ /    \
*    |    \| |/__ \| __|| |/   \ \__'  |\__'  |
*    |  .  | |  .  |__ || |  .  | __|  | __|  |
*    \____/|_|\___/|___||_|\___/ |____/ |____/
*
************************************************
*              github.com/blasio99
*
************************************************/

/*
* Problem 1 - Naive prime number search
*/

#include <stdio.h>
#include <time.h>

#define TRUE  1u
#define FALSE 0u

/* Fail macro to print error message */
#define FAIL() \
        do \
        { \
            printf("A is a bigger number than B"); \
        } while (0);
```

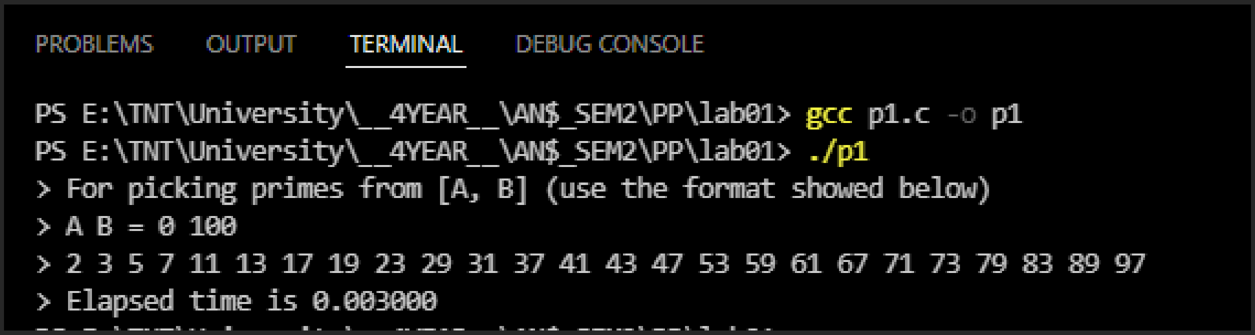
```
/* Assert test macro */
#define _assert(test) \
        do \
        { \
            if (!(test)) \
            { \
                FAIL(); \
                return ; \
             } \
        } while(0)

typedef unsigned char boolean;

boolean isPrime(unsigned int number);
void findAllPrimes(unsigned int A, unsigned int B);


int main(){

    clock_t clock_st;
    double elapsed_time;

    /* ---------- call for the problem ---------- */
    unsigned int A = 0;
    unsigned int B = 0;

    printf("> For picking primes from [A, B] (use the format showed below)\n> A B
= ");
    scanf("%d %d", &A, &B);

    clock_st = clock();

    findAllPrimes(A, B);
    /* ----- end operations for the problem ----- */

    elapsed_time = (double)(clock() - clock_st) / CLOCKS_PER_SEC;

    printf("> Elapsed time is %lf\n", elapsed_time);

}
```



```
boolean isPrime(unsigned int number){

    /* let's cover all cases */
    if (2 == number)
    {
        return TRUE;
    }
    else
    {
        /* go ahead */
    }

    if ((2 > number) || (0 == number % 2))
    {
        return FALSE;
    }
    else
    {
        /* go ahead */
    }

    for (unsigned int i = 3; (i * i) <= number; i += 2)
    {
        if (0 == number % i)
        {
            return FALSE;
        }
        else
        {
            /* continue execution */
        }
    }
    return TRUE;
}

void findAllPrimes(unsigned int A, unsigned int B)
{
    _assert(A < B);

    printf("> ");
```



```
    for (unsigned int i = A; i <= B; ++i)
    {
        if (TRUE == isPrime(i))
        {
            printf("%d ", i);
        }
        else
        {
            /* do nothing */
        }
    }
}
```

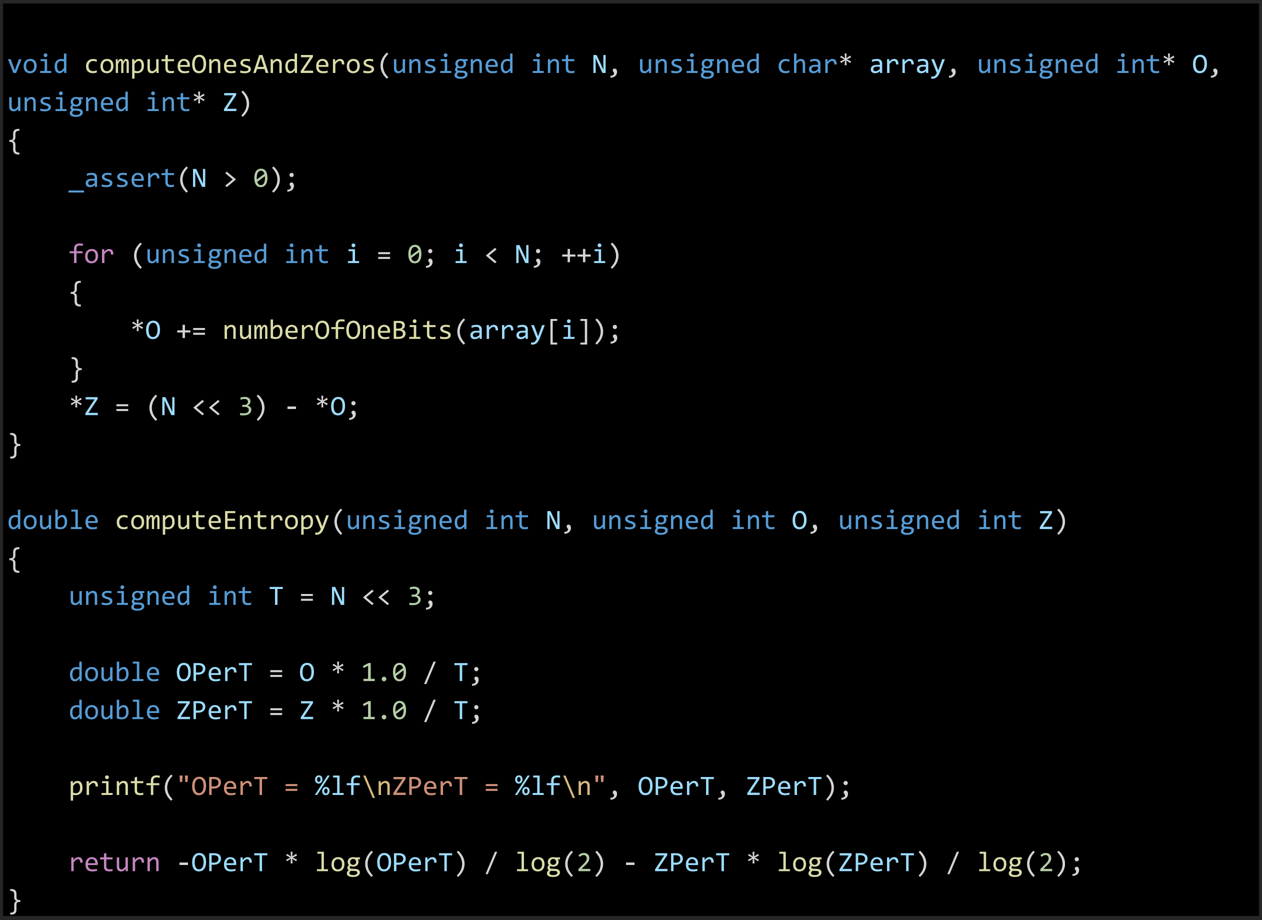


```
PS E:\TNT\University\__4YEAR__\AN$_SEM2\PP\lab01> gcc p1.c -o p1
PS E:\TNT\University\__4YEAR__\AN$_SEM2\PP\lab01> ./p1
> For picking primes from [A, B] (use the format showed below)
> A B = 0 100
> 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
> Elapsed time is 0.003000
```

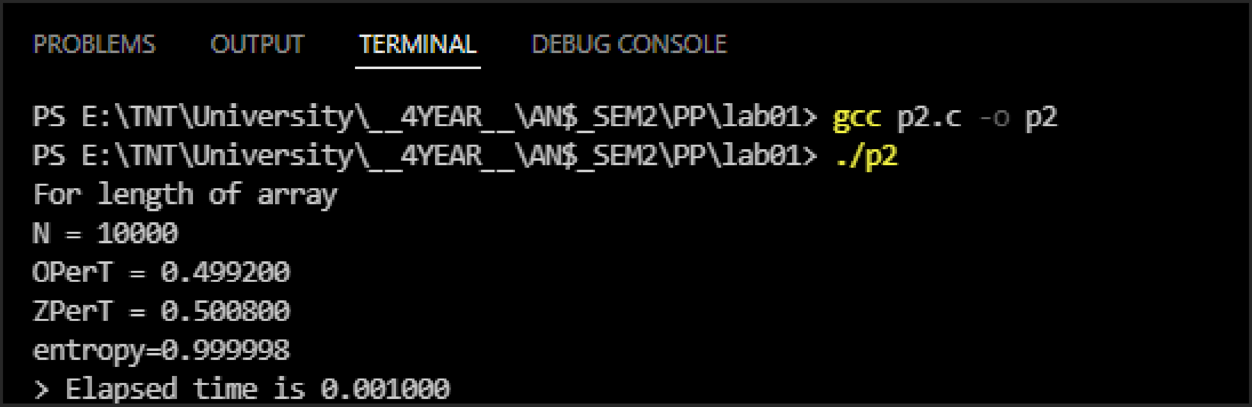
Problem 2

```c
/*
* Problem 2 - Compute the entropy of a binary random sequence
*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#define TRUE  1u
#define FALSE 0u

/* Fail macro to print error message */
#define FAIL() \
        do \
        { \
            printf("N is smaller than 0"); \
        } while (0);

/* Assert test macro */
#define _assert(test) \
        do \
        { \
            if (!(test)) \
            { \
                FAIL(); \
                return ; \
             } \
        } while(0)

typedef unsigned char boolean;

unsigned int numberOfOneBits(unsigned char number);
void computeOnesAndZeros(unsigned int N, unsigned char* array, unsigned int* O,
unsigned int* Z);
double computeEntropy(unsigned int N, unsigned int O, unsigned int Z);
```

```
int main(){

    clock_t clock_st;
    double elapsed_time;
    /* ---------- call for the problem ---------- */
    srand(time(NULL));

    unsigned int N = 0;
    unsigned int O = 0;
    unsigned int Z = 0;

    printf("For length of array\nN = ");
    scanf("%d", &N);

    clock_st = clock();
    unsigned char* S = (unsigned char*)malloc(sizeof(unsigned char) * N);

    for (unsigned int i = 0; i < N; ++i)
    {
        S[i] = rand() % 256;
    }
    computeOnesAndZeros(N, S, &O, &Z);

    printf("entropy=%lf\n", computeEntropy(N, O, Z));
    /* ----- end operations for the problem ----- */

    elapsed_time = (double)(clock() - clock_st) / CLOCKS_PER_SEC;

    printf("> Elapsed time is %lf\n", elapsed_time);

}

unsigned int numberOfOneBits(unsigned char number)
{
    unsigned int count = 0;

    for (int i = 0; i < 8; ++i)
    {
        count += (number & (1 << i)) >> i;
    }
    return count;
}
```



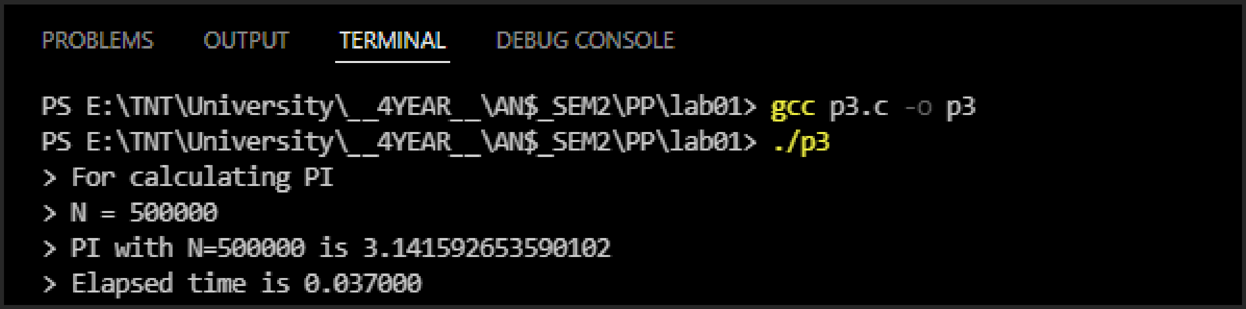
```c
void computeOnesAndZeros(unsigned int N, unsigned char* array, unsigned int* O,
unsigned int* Z)
{
    _assert(N > 0);

    for (unsigned int i = 0; i < N; ++i)
    {
        *O += numberOfOneBits(array[i]);
    }
    *Z = (N << 3) - *O;
}

double computeEntropy(unsigned int N, unsigned int O, unsigned int Z)
{
    unsigned int T = N << 3;

    double OPerT = O * 1.0 / T;
    double ZPerT = Z * 1.0 / T;

    printf("OPerT = %lf\nZPerT = %lf\n", OPerT, ZPerT);

    return -OPerT * log(OPerT) / log(2) - ZPerT * log(ZPerT) / log(2);
}
```



```
PS E:\TNT\University\__4YEAR__\AN$_SEM2\PP\lab01> gcc p2.c -o p2
PS E:\TNT\University\__4YEAR__\AN$_SEM2\PP\lab01> ./p2
For length of array
N = 10000
OPerT = 0.499200
ZPerT = 0.500800
entropy=0.999998
> Elapsed time is 0.001000
```

Problem 3

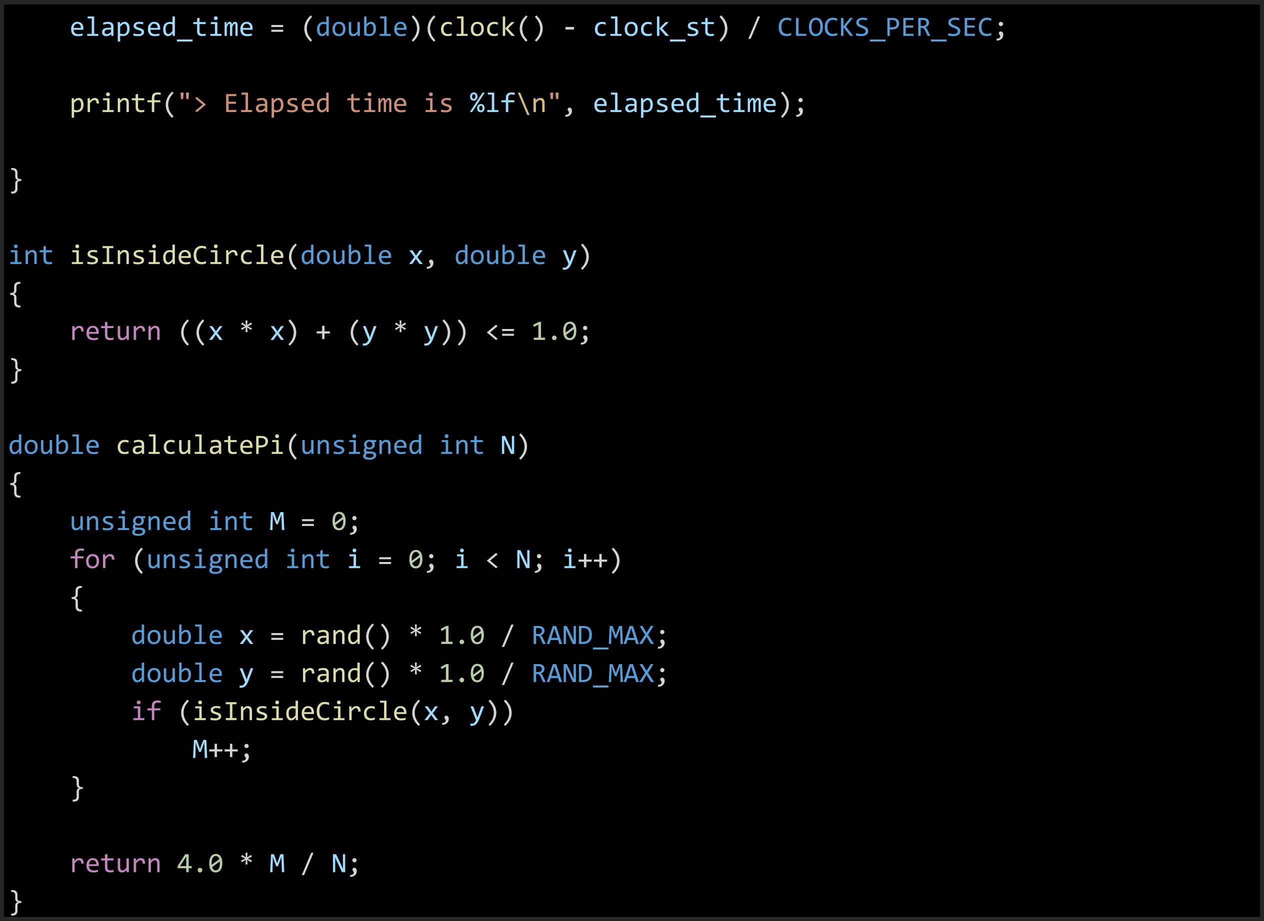
```c
/*
* Problem 3 – Compute the value of PI (π)
*/

#include <stdio.h>
#include <time.h>
#include <math.h>

double computePi(long N);

int main(){

    clock_t clock_st;
    double elapsed_time;

    /* ---------- call for the problem ---------- */
    long N = 0;

    printf("> For calculating PI\n> N = ");
    scanf("%d", &N);

    clock_st = clock();

    printf("> PI with N=%d is %0.15lf\n", N, computePi(N));

    /* ----- end operations for the problem ----- */

    elapsed_time = (double)(clock() - clock_st) / CLOCKS_PER_SEC;

    printf("> Elapsed time is %lf\n", elapsed_time);
}
double computePi(long N)
{
    double PI = 0;
    for (long i = 0; i < N; i++)
    {
        PI += 4.0 / (1 + pow(((i + 0.5) / N), 2));
    }
    return PI / N;
}
```



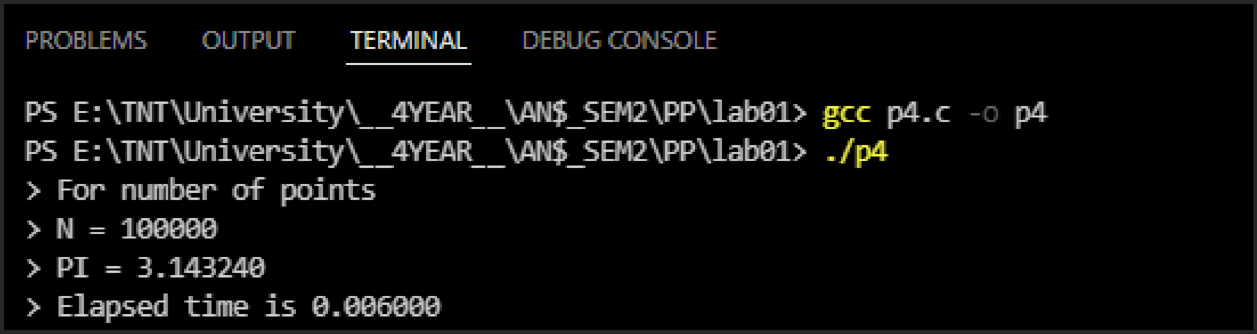
```
PS E:\TNT\University\__4YEAR__\AN$_SEM2\PP\lab01> gcc p3.c -o p3
PS E:\TNT\University\__4YEAR__\AN$_SEM2\PP\lab01> ./p3
> For calculating PI
> N = 500000
> PI with N=500000 is 3.141592653590102
> Elapsed time is 0.037000
```

Problem 4

```
/*
 * Problem 4 - Compute the value of PI (π) using the Monte Carlo method
 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

int isInsideCircle(double x, double y);
double calculatePi(unsigned int N);

int main(){

    clock_t clock_st;
    double elapsed_time;

    /* ---------- call for the problem ---------- */
    srand(time(NULL));

    unsigned int N = 100000000;

    printf("> For number of points\n> N = ");
    scanf("%d", &N);

    clock_st = clock();

    printf("> PI = %lf\n", calculatePi(N));
    /* ----- end operations for the problem ----- */
```



```c
    elapsed_time = (double)(clock() - clock_st) / CLOCKS_PER_SEC;

    printf("> Elapsed time is %lf\n", elapsed_time);

}

int isInsideCircle(double x, double y)
{
    return ((x * x) + (y * y)) <= 1.0;
}

double calculatePi(unsigned int N)
{
    unsigned int M = 0;
    for (unsigned int i = 0; i < N; i++)
    {
        double x = rand() * 1.0 / RAND_MAX;
        double y = rand() * 1.0 / RAND_MAX;
        if (isInsideCircle(x, y))
            M++;
    }

    return 4.0 * M / N;
}
```



```
PS E:\TNT\University\__4YEAR__\AN$_SEM2\PP\lab01> gcc p4.c -o p4
PS E:\TNT\University\__4YEAR__\AN$_SEM2\PP\lab01> ./p4
> For number of points
> N = 100000
> PI = 3.143240
> Elapsed time is 0.006000
```