```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

/*
Nome: Felipe Braz Marques
Matrícula: 22.1.4030

Como executar o código: gcc qA.c -o main -Wall
                        ./main

A análise assintótica pode ser feita da seguinte forma:
T(1) = 1
T(n) = 2T(n/2) + 7
T(n) = 2T(n/2) + O(1)

Pelo teorema mestre:
a = 2, b = 2, d = 0
log2 2 > 0

findMaxValue() = O(n)
*/

void findMaxValue(int* vector, int left, int right, int* maximum)
{
    if (right <= left)                            //1
        return;                                   //1

    int middle = (left + right) / 2;              //1
    if (vector[middle] > *maximum)                //2
    {
       *maximum = vector[middle];                 //2
    }

    findMaxValue(vector, left, middle, maximum);       // T(n/2)
    findMaxValue(vector, middle + 1, right, maximum);  // T(n/2)
}

int main()
{
    int vector[] = {7, 9, 5, 8, 11, 15};
    int length = sizeof(vector) / sizeof(vector[0]);
    int maximum = vector[0];

    findMaxValue(vector, 0, length, &maximum);

    printf("The highest value in the vector is: %d\n", maximum);

    return 0;
}

/*

Output:
The highest value in the vector is: 15

*/
```

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

/*
Nome: Felipe Braz Marques
Matrícula: 22.1.4030

Como executar o código: gcc qB.c -o main -Wall
                        ./main

A análise assintótica pode ser feita da seguinte forma:
T(1) = 1
T(n) = 2T(n/2) + 7
T(n) = 2T(n/2) + O(1)

Pelo teorema mestre:
a = 2, b = 2, d = 0
log2 2 > 0

findMaxMinValues() = O(n)
*/

void findMaxMinValues(int* vector, int left, int right, int* maximum, int* minimum)
{
    if (right <= left)                                   //1
        return;                                          //1

    int middle = (left + right) / 2;                     //1

    if (vector[middle] > *maximum)                       //1
    {
       *maximum = vector[middle];                        //1
    }
    if (vector[middle] < *minimum)                       //1
    {
        *minimum = vector[middle];                       //1
    }

    findMaxMinValues(vector, left, middle, maximum, minimum);         // T(n/2)
    findMaxMinValues(vector, middle + 1, right, maximum, minimum);    // T(n/2)
}

int main()
{
    int vector[] = {7, 9, 5, 8, 11, 15};
    int length = sizeof(vector) / sizeof(vector[0]);
    int maximum = vector[0];
    int minimum = 100000000;

    findMaxMinValues(vector, 0, length, &maximum, &minimum);

    printf("The highest value in the vector is: %d\n", maximum);
    printf("The lowest value in the vector is: %d\n", minimum);
    return 0;
}

/*
Output:
The highest value in the vector is: 15
The lowest value in the vector is: 5

*/
```

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

/*
Nome: Felipe Braz Marques
Matrícula: 22.1.4030

Como executar o código: gcc qC.c -o main -Wall
                        ./main

A análise assintótica pode ser feita da seguinte forma:
T(1) = 1
T(n) = T(n/2) + 7
T(n) = T(n/2) + O(1)

Pelo teorema mestre:
a = 1, b = 2, d = 0
log2 1 > 0

exponentiation() = O(log(n))
*/

int exponentiation(int base, int power)
{
    if (power == 0)                                         // 1
        return 1;                                           // 1

    int result = exponentiation(base, power / 2);           // T(n/2)

    if (power % 2 == 0)                                     // 2
        return result * result;                             // 2
    else
        return base * result * result;                      // 3
}

int main()
{
    int base = 2;
    int power = 5;
    int result = exponentiation(base, power);

    printf("%d raised to the power of %d is equal to %d\n", base, power, result);

    return 0;
}

/*
Output:
2 raised to the power of 5 is equal to 32

*/
```