**Exercise 5**

*Is an unsigned value odd or even?*

For this exercise we defined a main function, reading an unsigned integral value from cin. Then six cout statements were provided. The listing is provided below:

#include <iostream>

#include <math.h>

using namespace std;

int toEvenOrNotToEven(int value)

{

if(value == 0)

return 1;

else if(value == 1)

return 0;

else if(value<0)

return toEvenOrNotToEven(-value);

else

return toEvenOrNotToEven(value-2);

}

int main() {

int value;

cin >> value;

cout << ( value % 2 == 1 ? "odd" : "even") << '\n';

cout << ( (value/2\*2) == value ? "even" : "odd") << '\n';

cout << ( (value & 1) == 1 ? "odd" : "even") << '\n';

cout << ( (( value >> 1) << 1 ) == value ? "even" : "odd") << '\n';

cout << ( toEvenOrNotToEven(value) == 0 ? "odd" : "even") << '\n';

cout << ( pow(-1,value) == -1 ? "odd" : "even") << '\n';

}

*With each statement provide a short sematic comment explaining why the expression correctly performs its task.*

We will shortly describe the expressions in the chronological order displayed above. The first expression uses the modulo operator. If the remainder after division by 2 equals 0 the number is odd, otherwise it is even. The second line uses the divide operator. After division the value will be rounded off to a natural number. Since even numbers are dividable by 2, they do not need to be rounded off. Therefore only numbers that change after division and multiplication by 2 will be odd. The third expression uses a bitwise operator. Only the lowest-order bit determines if a number is odd or even, since all other bits are multiples of two. The operator & is used to determine if that bit was even or odd. The fourth line uses the same bitwise approach. This time the right shift operator makes the last digit disappear. When we shift it back with a left shift operation, we can see if the number changed. If that is the case, the number is odd since the ‘odd’ bit was used. The fifth line uses the function that was written at the top of the code. It uses recursion to determine if the number is odd or even. It substracts 2 from the number until it is 0 or 1. If it is 0 the number is odd, and otherwise it is even. Finally, the last line uses a classic mathematical trick. If -1 to the power of the number is negative, the number is odd and otherwise it is even.