EXERCISES UNIT 3

1. Write several versions for checking if a number is even. Sort them from less efficient to more efficient. Repeat the exercise for checking if a number is divisible by 8.

2. In many processors, the cost of integer multiplication is greater than other basic operations with integers. Write a code for calculating x*5 faster than the standard multiplication, being x an unsigned integer.

3. Write a function for counting the number of bits set to 1 in an unsigned integer. Improve the efficiency of the function taking into account that it will be used for counting the number of bits set in a very long sequence (for example 10^9 bits).

4. How many bytes does each element of the following vector occupy? Prove your answer.

```
class c {
  bool a;
  float b;
  unsigned short c;
};

c v[10];
```

5. The following function implements a sequential search in a vector. Make optimizations in the code. The optimizations may consist in improvements of the algorithm or in adjustments of the code. For each optimization check the execution time using the Linux command *time*. For this purpose, the program should contain an outer loop in order to repeat the function as many times as necessary.

```
// Return the position of the element.
// Out of bound if the element doesn't exist.
unsigned search(vector<int> &v, int x)
{
  unsigned pos = 0;
  while(pos < v.size() && v[pos] != x)
    pos++;
  return pos;
}
```

6. The following function in C++ evaluates a polynomial of grade n in the point x. Make optimizations in the code. The optimizations may consist in improvements of the algorithm or in adjustments of the code. For each optimization check the execution time using the Linux command *time*. For this purpose, the program should contain an outer loop in order to repeat the function as many times as necessary.

$a_{n-1}x^{n-1} + \ldots + a_2x^2 + a_1x + a_0$

```
float eval(const vector<float> &a, float x)
{
  float y;
  y = a[0];
  for(unsigned i = 1; i < v.size(); i++)
    y = y + a[i] * pow(x,i);
  return y;
}
```

7. Optimize the following loops:

a)

```
int i, j, k;
float  x[N], y[N];

for(k = 0; k < ITER; k++)
    for(i = 0; i < N; i++)
       if(i == 0)
           x[i] = 0;
       else if(i == N-1)
           x[i] = N-1;
       else
           x[i] = x[i] + y[i];
```

b)

```
int i, j, k;
float a, x[N], y[N];

for(k = 0; k < ITER; k++)
    for(i = 0; i < N; i++)
       if(a == 0.0)
           x[i] = 0;
       else
           y[i] = x[i] * y[i];
```

c)

```
int i, j, k;
float x[N], y[N];
float a;

for(k=0; k<ITER; k++){
    a=0.0;
    for(i=0; i<N; i++)
        a = a + x[i] * y[i];
    }
```