Bit Reader/Writer

The purpose of the first laboratory is to implement a bit reader/writer. On computer the smallest unit to store or read data is the byte and because in the following laboratories we will need the ability to write/read at bit level. But because on the computer we can't read/write at bit level we are going to mimic this behavior through a buffer (read buffer or write buffer). Thus you will have to implement to class – reading from a file and writing to a file. The use cases of this classes are presented below.

Beside the suggested approach you can come with your own solution/implementation. You can rename the methods or you can add whatever other fields/methods you fill necessary. Also you can implement reading/writing from the left side of a byte – or from the right hand side of the byte. It doesn't matter. It does matter that what you write as first bit, when you read it – it represents the same thing

The import item is that the following 4 methods to be implemented:

- BitReader
  - ReadBit()
  - ReadNBit(n)
- BitWriter
  - WriteBit (bit)
  - WriteNBit(numberOfBits, value)

The pseudocode for the 2 classes below is presented:

```
class BitReader
{
    private BufferReader; // The type  can be a byte or an array of items(each
element in the array will represent a BIT)
    private int NumberOfReadBits;

    public BitReader(string filePath) // filePath = the file path to the input
file
    {
        NumberOfReadBits = 0 ;
    }


    private bool IsBufferEmpty() {
        NumberOfReadBits == 0;
    }

    //The return type can be byte or int or even a boolean. Always this methods
returns 1 bit - 0 or 1
    private int ReadBit()
    {
```

```
        if(IsBufferEmpty())
        {
            //Read 1 byte from input file and put in inside BufferReader
            NumberOfReadBits = 8; // If I have filled in my Buffer than it means
I have 8 available bits
        }

        int result = 0;
        //take 1 bit from the buffer and put in into result
        //Probably decrease number of available bits
        NumberOfReadBits--;

        return result;
    }

    public uint ReadNBits(int nr) //nr will be a value [1..32]
    {
        uint result = 0;
        for(var i=nr-1;i>=0;i--)
        {
            // bit = ReadBit();
            // add bit to result
        }

        return result;
    }

}

class BitWriter
{
    private BufferWrite; // It can be a byte or an array of itemses(each element
in the array will represent a BIT)
    private int NumberOfBitsWrite;

    public BitWriter(string filePath) // file path of the output file
    {
        NumberOfBitsWrite = 0;
    }

    private bool IsBufferFull()
    {
        return NumberOfBitsWrite == 8;
    }
```

```
    //The value type can be byte or int or even a boolean. But always it will
take only the last bit of the value
    private void WriteBit(int value)
    {
        //scrie last bit from value into BufferWrite
        NumberOfBitsWrite++;

        if(NumberOfBitsWrite == 8)
        {
            NumberOfBitsWrite=0;
            //Write BufferWrite into the output file
        }
    }

    public void WriteNBits(int nr, int value) //nr will be a value [1..32]. Value
must be an unsigned number which can be store at least on 32 bits. E.g. in C#
UINT32
    {
        for(...)
        {
            // WriteBit(...)
        }

    }
}


class TestReaderWriter
{
    public void Test()
    {
        long NBR = 0;
        //deschide fisier sursa. Create output file
        //NBR  = 8 * InputFileSizeInBytes
        do
        {
            int nb;
            //nb = Random between [1..32]
            if(nb  > NBR)
            {
                nb = NBR;
            }

            uint value = BitReader.ReadNBits(nb);
            BitWriter.WriteNBits(nb, value);
```

```
            NBR-=nb;
        }while(NBR > 0);

        //Here both input and output should be the same
    }
}
```

Performance improvement

The above algorithm can be improved – from performance point of view. For e.g. let's take the following situation:

- BufferWriter
    - My buffer is empty
    - I want to write a value on 6 bits -> we can use the standard algorithm
    - After I want to write a value on 22 bits than the following can be done
        - I take 2 bits from the input value and write it in order to ensure that buffer is filled and will be dumped to the filesystem.
        - I have 20 bits remaining – in order not to call WriteBit for each of them –I can group them in groups of 8 bits and call – the default WriteByte method (each language/framework supports reading/writing of a single byte). Thus I will write 16 bits using the language specific WriteByte method. I still remain with 4 bits for which I will call the WriteBit method and use the BufferWriter.
        - Same logic can be applied for BufferReader