Exo 1

Q1. According to Frama-C RTE 4 runtime errors could happen:
1. a memory error when executing T[x]=y if x<0
2. a memory error when executing T[x]=y if x>=5
3. an integer overflow when executing x=x+1
4. an integer overflow when executing y=y+100

Q2. Running Frama-C EVA we get the following results:
- EVA is not able to prove that error 2 won't occur
- EVA is not able to prove that error 4 won't occur

The 2 other errors are discharged (they will definitely not occur)

Q3.

Running a VSA by hand we get the following results at the entry of each babsic block:

* without widening, the loop is unrolled up to termination:
(note that variable y is *not* constrained by condition of block B1)

Entering B0
x=bot, y=bot
Entering B1
x=[0,6], y=[0,600]
Entering B2
x=[0,5], y=[0,600]
Entering B3
x=[1,5], y=[0,600]
Entering B4
x=[0,6], y=[0,600]
Entering B5
x=[6,6], y=[0,600]

We get the same conclusion than Frama-C for error 2, but not for error 4 (no integer overflow when incrementing y)

* with widening/narrowing, variables are set to +infty after one iteration:
(note that variable y is *not* narrowed by condition of block B1)

Entering B0
x=bot, y=bot
Entering B1
x=[0,6], y=[0,+infty]
Entering B2
x=[0,5], y=[0,+infty]
Entering B3
x=[1,5], y=[0,+infty]
Entering B4
x=[0,6], y=[0,+infty]
Entering B5
x=[6,6], y=[0,+infty]

We get the same conclusions than Frama-C.

* According to the normal program execution, error 1 may really occur at runtime,
but error 2 will not occur. This last error is therefore a false positive.
(Frama-C is not able to discharge because it cannot catch the implicit relation between x and y,
which is caught without using widening/narrowing).

Q4. For N=1000 we would get:

```
Entering B0
      x=bot, y=bot
Entering B1
      x=[0,1001], y=[0,+infty]
Entering B2
      x=[0,1000], y=[0,+infty]
Entering B3
      x=[1,999], y=[0,+infty]
Entering B4
      x=[0,1000], y=[0,+infty]
Entering B5
      x=[1001,1001], y=[0,+infty]
```

Here error 2 is discharged, because within B3 we have x<1000.
However, error 4 is not discharged, and it is still a false positive ...

Q5.

We want to check under which conditions on N we would get a buffer overflow at line 11.
To do so we need to express a constraint on N (considered as a symbolic value) telling whether line 11 can be executed with x<0 or x>=N.
In practice, since the value of N impacts the number of loop iterations, we need to enumerate several values of N
(since each of them leads to a different path predicate for reaching line 11).

For instance, we should consider the following constraints:

```
  - no iteration
      N=0 and x=0 and x<N+1 and x%2=1 and (x<0 or x>=N),
      which is not satisfiable
  - 1 iteration
      N=1 and x=0 and x<N+1 and x%2=1 and (x<0 or x>=N) and
      x1=x+1 and x1<N+1 and x1%2=1 and (x1<0 or x1>=N),
            which is satisfiable since x1=1.
  - 2 iterations
      N=2 and x=0 and x<N+1 and x%2=1 and (x<0 or x>=N) and
      x1=x+1 and x1<N+1 and x1%2=1 and (x1<0 or x1>=N) and
      x2=x1+1 and x2<N+1 and x2%2=1 and (x2<0 or x2>=N)
        which is not satisfiable
  - etc.
```

In conclusion the best we can do here is to check a *finite* set of constraints (corresponding to each numbers of iterations), and we will get an error whenever N is odd.
But we won't be able to conclude for *any* value of N ...