**Study 1 codes**

The following code snippets are presented to the volunteers without indicating the bugs' locations. In this file, we show them for reproducibility purposes. Some of the bugs are positioned in one line of code, while others occupy more than one line of code.
The codes below are annotated by the cyclomatic complexity metric VG and divided into coherent non-overlapping regions.

```
Fibo ->Bucket sort -> 41-60 + 61-71...80 + 81-100                Task VG = 10 = Complex++
41    void bucketSort(int numb, int size, int array[size], int * res)
42    {
43        int bucks[numb][size];
44        int szbucks[numb];
45        int bi,bpos, i,j,aux, max, bwidth;
46        max = 0;                   BUG B1                          VG = 4 = Medium
47        for (i=1; i< size; i++)
48            if (max<array[i])
49                max=array[i];
50        bwidth = 1 + max / numb;     BUG B2
51        for (i = 0; i< max; i++)
52            szbucks[i] = 0;
53        for (i=0; i<size; i++)                                     VG = 2 = Simple
54        {
55            bi = array[i]/bwidth;
56            bpos = szbucks[bi];
57            bucks[bi][bpos] = array[i];
58            szbucks[bi]++;
59        }
60
61        for (bi=0; bi < numb; bi++)                                VG = 5 = Complex
62        {
63            for (i = 0; i<szbucks[bi]-1; i++)
64                for (j=0; j<szbucks[bi]-i-1; j++)
65                    if (bucks[bi][i] > bucks[bi][i+1])     BUG B3
66                    {
67                        aux = bucks[bi][j];
68                        bucks[bi][j] = bucks[bi][j+1];
69                        bucks[bi][j+1] = aux;
70                    }
71        }
                                                                     VG = 3 = Medium
81        bi = 0;     BUG B4
82        while (bi<numb)    BUG B4
83        {
84            for (j = 0; j<szbucks[bi]; j++)
85            {
86                res[i] = bucks[bi][j];
87                i++;
88            }
89            bi++;
90        }
91    }
```

Fibo -> 21-29...40 **Task VG = 2 = Simple**

```
21    unsigned int fibo(unsigned int n)
22    {                                                       VG = 2 = Simple
23        unsigned int res;
24        if (n == 1)                                                   BUG F1
25            res = 1;                                                  BUG F1
26        else                                                          BUG F1
27            res = fibo(n - 1) + fibo(n - 2);
28        return res;
29    }
```

***(The entire task is just one area)***

```
Hondt -> 21-40 + 41-52...60                                 Task VG = 5 = Complex

21  void hondt(int votes[], int seats[], int num_parties, int num_seats)
22  {
23          int seats_allocated;
24          double quotients[num_parties];
25          int i, max_i;
26          double max;
27          i = 0;                                   BUG H1   VG = 1 = Simple
28          seats_allocated = 0;                     BUG H1
29          while(seats_allocated < num_seats)       BUG H1   with sub-area VG = 3 = Medium
30          {                                      BUG H2   without VG also = 3 = Medium
31                  while(i < num_parties)         BUG H2    ** both excluding next area **
32                  {                                        *(breaks the while loop syntax)*
33              double quotient = votes[i] / seats[i];  BUG H3    sub-area VG = 1 = Simple
34                          quotients[i] = quotient;
35                          i++;
36                  }
37                  max = quotients[0];
38                  max_i = 0;
39                  i = 1;
40                                                                    VG = 3 = Medium
41                  while(i < num_parties)
42                  {
43                          if(quotients[i] >= max)
44                          {
45                                  max = quotients[i];
46                                  max_i = i;
47                          }
48                          i++;
49                  }
50                  seats_allocated++; BUG H4
51          } BUG H4
52  }
```

Matdeti -> 21-51...60 + 61-79...80 + 81-88...100 Task VG = 10 = Complex++

```
41      int mdeterminant(int size, int mat[size][size])
42      {
43          int det, subm, l, c, ls, cs, part;                              VG = 4 = Medium
44          int submat[size][size-1][size-1], coefs[size];
45          if (size < 1)
46              return 0;
47          if (size == 1)
48              return mat[0][0];
49          if (size == 2)
50              return mat[0][0]*mat[1][1] - mat[0][1]*mat[1][0];
51          subm=0;

61          while (subm<size) {              with sub-area VG = 4 = Medium / without VG = 3
62              ls = 0;
63              l = 1;
64              while (l<size) {
65                  cs=0;
66                  c = 0;
67                  while (c<size) {                                   Sub-area: VG = 2 = Simple
68                      submat[subm][ls][cs] = mat[l][c];   BUG M1
69                      cs++; BUG M2
70                  } BUG M2
71                  l++;  BUG M3
72              }
73              subm++;
74          }
75          for (subm=0;subm<size;subm++)                                   VG = 3 = Medium
76              if (subm==0)
77                  coefs[subm] = 1;
78              else
79                  coefs[subm] = coefs[subm-1];       Bug M4
80
81          det = 0;                                                        VG = 2 = Simple
82          part=0;
83          while (part<size) {
84              det += coefs[part]*mat[0][part]*mdeterminant(size-1,submat[part]);
85              part+=1;
86          }
87          return det;
88      }
```