Context

The regions are identified in the following manner:

- There are three programs, C1, C2, C3;
- Each program may have 1 or more functions (methods);
- There are two or more levels of regions: top level regions, and nested regions;
- Top-level regions are identified as *program.region-id*. Program is C1, C2 or C3: *region-id* is a letter A, B,C, etc. E.g., region C1.B is the second region of program C1;
- Nested regions are identified by appending a sequential number to the containing top-level region. E.g., C2.3.B is a nested region contained inside top-level region C2.B.
- In some cases (large top-level regions), several letters were used for the ID of a top-level regions instead of just one letter. In these cases, if the region is further split, the first nesting level is one of the letter used in the top-level, and the second level of nesting uses a number. In all other cases, splitting a top-level region having a single letter as ID results in sub-regions with number as ID.
- Lines numbers are given in this document for ease of reading. Note that these line numbers are specific to this document and were not given in the actual experiment (this means that Eye tracking data processing is not related to these numbers here)

The code is presented visually in tables with the following format

Region ID	Line number Code

For each program a summary is also presented.

For clarity, each program starts at the top of a new page.

Efforts were made to avoid breaking table lines across page boundaries, but for program C3 that was unavoidable.

Program C1

Code of program C1

```
01 public static int getResult(int[] sequence, int lower, int upper) {
02     int result = 0;
03     if (sequence == null)
04         return result;
05     for (int n : sequence) {
06         if (n>=lower && n<=upper)
07             result++;
08     }
09     return result;
10 }
11 public static void main(String[] args) {
12    int[] sequence = {-7, 1, 5, 2, -4, 3, 0};
13    int result = getResult(sequence,2,4);
14    System.out.println("Result = " + result);
15 }
```

Top-level regions for program C1

C1.AB	`01 public static int getResult(int[] sequence, int lower, int upper) {`
	`02     int result = 0;`
	`03     if (sequence == null)`
	`04         return result;`
	`05     for (int n : sequence) {`
	`06         if (n>=lower && n<=upper)`
	`07             result++;`
	`08     }`
	`09     return result;`
	`10 }`

C1.C	`11 public static void main(String[] args) {`
	`12     int[] sequence = {-7, 1, 5, 2, -4, 3, 0};`
	`13     int result = getResult(sequence,2,4);`
	`14     System.out.println("Result = " + result);`
	`15 }`

Nested regions of program C1

C1.A	`01 public static int getResult(int[] sequence, int lower, int upper) {`
	`02     int result = 0;`
	`03     if (sequence == null)`
	`04         return result;`

C1.B	`05     for (int n : sequence) {`
	`06         if (n>=lower && n<=upper)`
	`07             result++;`
	`08     }`
	`09     return result;`
	`10 }`

The following table summarizes the code regions on Program C1:

Region ID	Start line	End line	Num. of lines	Comments
C1.A	01	04	4	Region of A of program C1
C1.B	05	10	6	Region of B of program C1
C1.C	11	15	5	Region of C of program C1

Program C2

Code of program C2

```java
private static byte[] getInts(String digs) {
    byte[] result = new byte[digs.length()];
    for (int i = 0; i < digs.length(); i++) {
        char c = digs.charAt(i);
        if (c < '0' || c > '9') {
            throw new IllegalArgumentException("Invalid string " + c
                    + " at position " + i);
        }
        result[digs.length() - 1 - i] = (byte) (c - '0');
    }
    return result;
}
public static String getResult(String num1, String num2) {
    byte[] left = getInts(num1);
    byte[] right = getInts(num2);
    byte[] result = new byte[left.length + right.length];
    for (int rightPos = 0; rightPos < right.length; rightPos++) {
        byte rightDigit = right[rightPos];
        byte temp = 0;
        for (int leftPos = 0; leftPos < left.length; leftPos++) {
            temp += result[leftPos + rightPos];
            temp += rightDigit * left[leftPos];
            result[leftPos + rightPos] = (byte) (temp % 10);
            temp /= 10;
        }
         int destPos = rightPos + left.length;
        while (temp != 0) {
            temp += result[destPos] & 0xFFFFFFFFL;
            result[destPos] = (byte) (temp % 10);
            temp /= 10;
            destPos++;
        }
    }
    StringBuilder stringResultBuilder = new StringBuilder(result.length);
    for (int i = result.length - 1; i >= 0; i--) {
        byte digit = result[i];
        if (digit != 0 || stringResultBuilder.length() > 0) {
            stringResultBuilder.append((char) (digit + '0'));
        }
    }
    return stringResultBuilder.toString();
}
public static void main(String[] args) {
    System.out.println(getResult("1234","56789"));
}
```

Top-level regions for program C2

C2.A

```java
private static byte[] getInts(String digs) {
    byte[] result = new byte[digs.length()];
    for (int i = 0; i < digs.length(); i++) {
        char c = digs.charAt(i);
        if (c < '0' || c > '9') {
            throw new IllegalArgumentException("Invalid string " + c
                    + " at position " + i);
        }
        result[digs.length() - 1 - i] = (byte) (c - '0');
    }
    return result;
}
```

C2.BCD

```java
public static String getResult(String num1, String num2) {
    byte[] left = getInts(num1);
    byte[] right = getInts(num2);
    byte[] result = new byte[left.length + right.length];
    for (int rightPos = 0; rightPos < right.length; rightPos++) {
        byte rightDigit = right[rightPos];
        byte temp = 0;
        for (int leftPos = 0; leftPos < left.length; leftPos++) {
            temp += result[leftPos + rightPos];
            temp += rightDigit * left[leftPos];
            result[leftPos + rightPos] = (byte) (temp % 10);
            temp /= 10;
        }
         int destPos = rightPos + left.length;
        while (temp != 0) {
            temp += result[destPos] & 0xFFFFFFFFL;
            result[destPos] = (byte) (temp % 10);
            temp /= 10;
            destPos++;
        }
    }
    StringBuilder stringResultBuilder = new StringBuilder(result.length);
    for (int i = result.length - 1; i >= 0; i--) {
        byte digit = result[i];
        if (digit != 0 || stringResultBuilder.length() > 0) {
            stringResultBuilder.append((char) (digit + '0'));
        }
    }
    return stringResultBuilder.toString();
}
```

C2.E

```java
public static void main(String[] args) {
    System.out.println(getResult("1234","56789"));
}
```

Nested regions of program C2 (level 1)

Resulting from splitting top-level region C2.BCD)

C2.B (from C1.BCD)

```java
public static String getResult(String num1, String num2) {
    byte[] left = getInts(num1);
    byte[] right = getInts(num2);
    byte[] result = new byte[left.length + right.length];
```

C2.C (from C1.BCD)

```java
    for (int rightPos = 0; rightPos < right.length; rightPos++) {
        byte rightDigit = right[rightPos];
        byte temp = 0;
        for (int leftPos = 0; leftPos < left.length; leftPos++) {
            temp += result[leftPos + rightPos];
            temp += rightDigit * left[leftPos];
            result[leftPos + rightPos] = (byte) (temp % 10);
            temp /= 10;
        }
         int destPos = rightPos + left.length;
        while (temp != 0) {
            temp += result[destPos] & 0xFFFFFFFFL;
            result[destPos] = (byte) (temp % 10);
            temp /= 10;
            destPos++;
        }
    }
```

C2.D (from C1.BCD)

```java
    StringBuilder stringResultBuilder = new StringBuilder(result.length);
    for (int i = result.length - 1; i >= 0; i--) {
        byte digit = result[i];
        if (digit != 0 || stringResultBuilder.length() > 0) {
            stringResultBuilder.append((char) (digit + '0'));
        }
    }
    return stringResultBuilder.toString();
}
```

Nested regions for program C2 (level 2)

Resulting from splitting region C2.C)

C2.C.1

```java
    for (int rightPos = 0; rightPos < right.length; rightPos++) {
        byte rightDigit = right[rightPos];
        byte temp = 0;
```

C2.C.2

```java
        for (int leftPos = 0; leftPos < left.length; leftPos++) {
            temp += result[leftPos + rightPos];
            temp += rightDigit * left[leftPos];
            result[leftPos + rightPos] = (byte) (temp % 10);
            temp /= 10;
        }
```

C2.C.3	`26         int destPos = rightPos + left.length;`
	`27         while (temp != 0) {`
	`28             temp += result[destPos] & 0xFFFFFFFFL;`
	`29             result[destPos] = (byte) (temp % 10);`
	`30             temp /= 10;`
	`31             destPos++;`
	`32         }`
	`33     }`

The following table summarises the code regions on Program C2:

Region ID	Start line	End line	Num, of lines	Comments
C2.A	01	12	12	Region of A of program C2
C2.B	13	16	4	Region of B of program C2
C2.C	17	33	17	Region of C of program C2
C2.D	34	42	9	Region of D of program C2
C2.E	43	45	3	Region of E of program C2
C2.C.1	17	19	3	Region C.1 nested inside region C2.C
C2.C.2	20	25	6	Region C.2 nested inside region C2.C
C2.C.3	26	33	8	Region C.3 nested inside region C2.C

Program C3

Code of program C2

```
static boolean verif(int co[][][], int ci[][][], int p[], int d[]) {
    if (co == null || ci == null || p == null || d == null)
        return false;
    if (p.length <3 || d.length <3)
       return false;
    if (co.length<ci.length || co[0].length<ci[0].length || co[0][0].length < ci[0][0].length)
        return false;
    int x,y,z;
    int a,b,c;
    int ma,mb,mc;
    p[0] = p[1] = p[2] = d[0] = d[1] = d[2] = 0;
    for (x=0; x<co.length - ci.length; x++)
        for (y=0; y<co[0].length - ci[0].length;y++)
            for (z=0; z<co[0][0].length - ci[0][0].length; z++) {
                ma = ci.length;
                mb = ci[0].length;
                mc = ci[0][0].length;
                for (a = 0; a<ma; a++) {
                    b = 0;
                    for (; b<mb; b++) {
                        c = 0;
                        for (; c<mc && co[x+a][y+b][z+c] == ci[a][b][c]; c++);
                        if (c == 0) {
                            mb = b;
                            break;
                        }
                        if (c < mc)
                            mc = c;
                    }
                    if (b==0) {
                        ma = a;
                        break;
                    }
                    if (b<mb)
                        mb = b;
                }
                if (ma*mb*mc > d[0]*d[1]*d[2]) {
                    p[0] = x; p[1] = y; p[2] = z;
                    d[0] = ma; d[1] = mb; d[2] = mc;
                }
            }
    return true;
}
```

Top-level regions for program C3

C3.A

```java
static boolean verif(int co[][][], int ci[][][], int p[], int d[]) {
    if (co == null || ci == null || p == null || d == null)
        return false;
    if (p.length <3 || d.length <3)
        return false;
    if (co.length<ci.length || co[0].length<ci[0].length || co[0][0].length < ci[0][0].length)
        return false;
```

C3.B

```java
    int x,y,z;
    int a,b,c;
    int ma,mb,mc;
    p[0] = p[1] = p[2] = d[0] = d[1] = d[2] = 0;
```

C3.C

```java
    for (x=0; x<co.length - ci.length; x++)
        for (y=0; y<co[0].length - ci[0].length;y++)
            for (z=0; z<co[0][0].length - ci[0][0].length; z++) {
                ma = ci.length;
                mb = ci[0].length;
                mc = ci[0][0].length;
                for (a = 0; a<ma; a++) {
                    b = 0;
                    for (; b<mb; b++) {
                        c = 0;
                        for (; c<mc && co[x+a][y+b][z+c] == ci[a][b][c]; c++);
                        if (c == 0) {
                            mb = b;
                            break;
                        }
                        if (c < mc)
                            mc = c;
                    }
                    if (b==0) {
                        ma = a;
                        break;
                    }
                    if (b<mb)
                        mb = b;
                }
                if (ma*mb*mc > d[0]*d[1]*d[2]) {
                    p[0] = x; p[1] = y; p[2] = z;
                    d[0] = ma; d[1] = mb; d[2] = mc;
                }
            }
    return true;
}
```

Nested regions for program C3

Resulting from slitting regions top-level C3.A and C3.C

Some of the nested regions resulting from C3.C overlap with one another. This was intentional and was aimed at the exploratory work involved in the study.

Regions from C3.A

C3.A.1

```
01 static boolean verif(int co[][][], int ci[][][], int p[], int d[]) {
02     if (co == null || ci == null || p == null || d == null)
03         return false;
```

C3.A.2

```
04     if (p.length <3 || d.length <3)
05         return false;
```

C3.A.3

```
06     if (co.length<ci.length || co[0].length<ci[0].length || co[0][0].length < ci[0][0].length)
07         return false;
```

Regions from C3.C

C3.C.1

```
13         for (y=0; y<co[0].length - ci[0].length;y++)
14             for (z=0; z<co[0][0].length - ci[0][0].length; z++) {
15                ma = ci.length;
16                mb = ci[0].length;
17                mc = ci[0][0].length;
18                for (a = 0; a<ma; a++) {
19                     b = 0;
20                     for (; b<mb; b++) {
21                         c = 0;
22                         for (; c<mc && co[x+a][y+b][z+c] == ci[a][b][c]; c++);
23                         if (c == 0) {
24                             mb = b;
25                             break;
26                         }
27                         if (c < mc)
28                             mc = c;
29                     }
30                     if (b==0) {
31                         ma = a;
32                         break;
33                     }
34                     if (b<mb)
35                         mb = b;
36                 }
37                 if (ma*mb*mc > d[0]*d[1]*d[2]) {
```

```
38              p[0] = x; p[1] = y; p[2] = z;
39              d[0] = ma; d[1] = mb; d[2] = mc;
40          }
41      }
```

C3.C.2

```
14      for (z=0; z<co[0][0].length - ci[0][0].length; z++) {
15          ma = ci.length;
16          mb = ci[0].length;
17          mc = ci[0][0].length;
18          for (a = 0; a<ma; a++) {
19              b = 0;
20              for (; b<mb; b++) {
21                  c = 0;
22                  for (; c<mc && co[x+a][y+b][z+c] == ci[a][b][c]; c++);
23                  if (c == 0) {
24                      mb = b;
25                      break;
26                  }
27                  if (c < mc)
28                      mc = c;
29              }
30              if (b==0) {
31                  ma = a;
32                  break;
33              }
34              if (b<mb)
35                  mb = b;
36          }
37          if (ma*mb*mc > d[0]*d[1]*d[2]) {
38              p[0] = x; p[1] = y; p[2] = z;
39              d[0] = ma; d[1] = mb; d[2] = mc;
40          }
41      }
```

C3.C.3

```
15          ma = ci.length;
16          mb = ci[0].length;
17          mc = ci[0][0].length;
```

C3.C.4

```
18          for (a = 0; a<ma; a++) {
19              b = 0;
20              for (; b<mb; b++) {
21                  c = 0;
22                  for (; c<mc && co[x+a][y+b][z+c] == ci[a][b][c]; c++);
23                  if (c == 0) {
24                      mb = b;
25                      break;
26                  }
27                  if (c < mc)
28                      mc = c;
29              }
30              if (b==0) {
31                  ma = a;
32                  break;
```

	`33         }`
	`34         if (b<mb)`
	`35             mb = b;`
	`36     }`

C3.C.5	`19         b = 0;`
	`20         for (; b<mb; b++) {`
	`21             c = 0;`
	`22             for (; c<mc && co[x+a][y+b][z+c] == ci[a][b][c]; c++);`
	`23             if (c == 0) {`
	`24                 mb = b;`
	`25                 break;`
	`26             }`
	`27             if (c < mc)`
	`28                 mc = c;`
	`29         }`

C3.C.6	`21             c = 0;`
	`22             for (; c<mc && co[x+a][y+b][z+c] == ci[a][b][c]; c++);`
	`23             if (c == 0) {`
	`24                 mb = b;`
	`25                 break;`
	`26             }`
	`27             if (c < mc)`
	`28                 mc = c;`
	`29         }`

C3.C.7	`21             c = 0;`
	`22             for (; c<mc && co[x+a][y+b][z+c] == ci[a][b][c]; c++);`

C3.C.8	`23             if (c == 0) {`
	`24                 mb = b;`
	`25                 break;`
	`26             }`

C3.C.9	`27             if (c < mc)`
	`28                 mc = c;`

C3.C.10	`30         if (b==0) {`
	`31             ma = a;`
	`32             break;`
	`33         }`

C3.C.11	`34         if (b<mb)`
	`35             mb = b;`

C3.C.12	`37     if (ma*mb*mc > d[0]*d[1]*d[2]) {`

```
38                      p[0] = x; p[1] = y; p[2] = z;
39                      d[0] = ma; d[1] = mb; d[2] = mc;
40                  }
```

The following table summarizes the code regions on Program C3:

Region ID	Start line	End line	Num. of lines	Comments
C3.A	01	07	7	Region of A of program C3
C3.A.1	01	03	3	Region A.1 nested inside region C3.A
C3.A.2	04	05	2	Region A.2 nested inside region C3.A
C3.A.3	06	07	2	Region A.3 nested inside region C3.A
C3.B	08	11	4	Region of B of program C3
C3.C	12	43	32	Region of C of program C3
C3.C.1	13	41	29	Region C.1 nested inside region C3.C
C3.C.2	15	41	27	Region C.2 nested inside region C3.C
C3.C.3	15	17	3	Region C.3 nested inside region C3.C
C3.C.4	18	36	19	Region C.4 nested inside region C3.C
C3.C.5	19	29	11	Region C.5 nested inside region C3.C
C3.C.6	21	29	9	Region C.6 nested inside region C3.C
C3.C.7	21	22	2	Region C.7 nested inside region C3.C
C3.C.8	23	26	4	Region C.8 nested inside region C3.C
C3.C.9	27	28	2	Region C.9 nested inside region C3.C
C3.C.10	30	33	4	Region C.10 nested inside region C3.C
C3.C.11	34	35	2	Region C.11 nested inside region C3.C
C3.C.12	37	40	4	Region C.12 nested inside region C3.C