This test is divided into two parts, a questionnaire with short questions (1-6), and a coding question (7). Please answer questions 1-6 with short answers, no need to write a proof of your answer. Question 7 should be answered with a C++ program that we should compile and run.

1) What is the complexity of the following two loops:
    a)

```
for (i = 0; i < N; ++i)
      for (j = 0; j < N; ++j);
```

    b)

```
for (i = 0; i < N; ++i)
      for (j = i; j < N; ++j);
```

2) Order the following operations from fast to slow:
    a) disk read
    b) register read
    c) context switch
    d) cache read
    e) ram read
    f) internet read

3) Read the following program:

```
int main() {
  int x = 1;
  static int y = 2;
  int *z = new int;
  z[0] = 3;
}
```

For each value below, is it stored in the stack? answer with “YES” or “NO”:
    a) x
    b) y
    c) z
    d) z[0]

4) For each sort algorithm, write which scenario matches it best.

   Algorithms:

   a) Quick sort
   b) Bubble sort
   c) Counting sort

   Scenarios:

   d) Need to sort data you know nothing about.
   e) Need to sort data that are almost completely sorted.
   f) Need to sort data made out of integers with a low distance between min and max.

5) What are the average case complexities in the following data structures for searching an element data structures:
   a) Array
   b) sorted array
   c) balanced binary tree (a.k.a. std::set)
   d) hash table (a.k.a. std::unordered_set)

6) Find a place and a reason why the program crashes:

```
struct A {
  char* str;
  A() : str(NULL) {}
  ~A() { delete[] str; }
}

void foo() {
  A a;
}

void bar(A a) {
  printf("%s", a.str);
}

int main() {
 foo();
 A a1;
 a1.str = new char[7];
```

```
 strcpy(a1.str, "Jessie");

 bar(a1);

 return 0;
}
```

7) Note: in this coding question, we are **not interested in object oriented design**, so you don't have to design complicated classes. Usage of STL is encouraged but not required.

   Write a method that finds shared edges between triangles

   A triangle is represented by 3 vertex IDs (v_0, v_1, v_2), which implies the triangle edges: $E_0 = (v_0, v_1)$, $E_1 = (v_1, v_2)$, $E_2 = (v_2, v_0)$.
   2 triangles share an edge if they share 2 vertex IDs, for example triangles $(0, 2, 7)$ and $(6, 2, 0)$ share the following edge $(0, 2)$.

   Your input is an array of vertex ids which will be interpreted 3 at a time,
   for example, 9 vertices "0,2,7,1,3,5,6,2,0" would represent 3 triangles with vertex IDs : $T_1 = (0, 2, 7)$, $T_2 = (1, 3, 5)$ and $T_3 = (6, 2, 0)$ and 9 corresponding edges.

   Lets define v_i as the vertex in index i in the input array and E_i as the edge defined by (v_i, v_{i+1}) if v_i and v_{i+1} represent the same triangle else E_i is defined by (v_i, v_{i-2})

   Your output will be an array. Cell i in the array will be set to -1 if edge E_i is not shared with other triangles and to j if it is shared with edge E_j. You may assume that each edge is shared at most once.

   For example, for input "0,2,7,1,3,5,6,2,0" only the edge $E_0 = (0, 2)$ and edge $E_7 = (0, 2)$ are shared between triangles, therefore the output array will be "7,-1,-1,-1,-1,-1,-1,0,-1".