Final Exam. 2020-12-17.

120 minutes.

Question 1 (Using STL).
There is a datatype `Student` to store student data. Students are not allowed to change their birthdates and their names while studying (`birth` shows how many days separate January 1, 1970 and their birthday).

You want to store these students in an STL data structure `unordered_set` that would allow to iterate over them (and also insert new students or remove them once they graduate).
This STL class requires that you implement a hashing function and an equals operator. In this task you will write your own hashing function (and also evaluate, if other hash functions are good for this task).

(A) Write your own hash function for the class `Student`. It should use exactly $\overline{abc} + 1$ hash buckets, where a, b, c are the last 3 digits of your Student ID. (For example, if your Student ID ends with `011`, then you should use 12 buckets.)
Your hashfunction expression can use any of the following:

1. Five integer arithmetic operators `a` + `b`, `a` − `b`, `a` ∗ `b`, `a`/`b`, `a`%`b`.
2. Six kinds of bitwise operators; see `https://bit.ly/37pXihp` (AND, OR, XOR, NOT, left shift and right shift).
3. `std::hash` functions for the types `int`, `double`, `string` (see examples below).

(B) Find, if any of the following expressions can be used as hash functions for the struct type `Student`, if substituted in Line 24. If they are bad, specify the reason why.

```
/* EXPR 1 */
(st.birth % 7) + st.name.length();
```

```
/* EXPR 2 */
hash<int>{}(st.birth) +
hash<double>{}(st.weight);
```

```
/* EXPR 3 */
hash<int>{}(st.birth) ^
hash<string>{}(st.string);
```

```
/* EXPR 4 */
hash<int>{}(st.birth) &
hash<string>{}(st.string);
```

Here is the code to surround hash functions in this problem (see placeholder on Line 24).

```
1  #include <iostream>
2  #include <unordered_set>
3  #include <string>
4
5  using namespace std;
6
7  struct Student {
8    int birth;
9    double weight;
10   string name;
11 };
12
13 bool operator== (Student const& lhs,
14   Student const& rhs) {
15   return (lhs.birth == rhs.birth) &&
16          (lhs.name == rhs.name);
17 }
18
19 // wrap the hash functor
20 struct Hash {
21   size_t operator()
22       (const Student &st) const {
23     unsigned hashValue =
24       /* YOUR EXPRESSION HERE */;
25     cout <<"h="<<hashValue<<endl;
26     return hashValue;
27   }
28 };
29
30 int main() {
31   unordered_set<Student,Hash> set;
32   Student s1 = {12, 3.14, "Alex"};
33   Student s2 = {13, 6.E23, "Dina"};
34   set.insert(s1);
35   set.insert(s2);
36 }
```

Question 2 (Time Complexity).

	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
v_0	–	1	0	0	0	1	0	0	0	0
v_1	1	–	0	0	0	0	0	1	1	0
v_2	0	0	–	1	0	0	0	0	1	0
v_3	0	0	1	–	0	1	0	0	0	0
v_4	0	0	0	0	–	1	0	0	0	1
v_5	1	0	0	1	1	–	1	0	0	0
v_6	0	0	0	0	0	1	–	1	0	0
v_7	0	1	0	0	0	0	1	–	0	1
v_8	0	1	1	0	0	0	0	0	–	0
v_9	0	0	0	0	1	0	0	1	0	–

(A) Assume that a, b, c are the last 3 digits of your Student ID. Find $(v_c, v_d) \in E$ – the smallest integer $d \neq c$ for which there is an edge (v_c, v_d) in the graph defined by the adjacency matrix above.
Run the BFS traversal algorithm on the graph G' that remains, if you remove edge (v_c, v_d) from the original graph. Order the vertices by their level in the BFS-tree. Show the BFS discovery edges bold (or different color) so that they are distinguishable from the cross edges.

(B) Find, if there is a path from v_c to v_d in G' (where the edge (v_c, v_d) is removed). If there is such a path, find its length.

(C) Consider the following pseudocode to find the shortest cycle in a graph $G = (V, E)$. Assume that the graph G is a large one, and it has n vertices ($|V| = n$) and m edges ($|E| = m$). Describe the time complexity of this pseudocode in terms of variables m and n. Use the Big-O notation.
Namely, suggest the best (the slowest growing) $f(m, n)$ such that the total number of steps (used for this algorithm itself and also for all the BFS traversals used by it) is $O(f(m, n))$ in the worst case. Explain your answer.

```
    ShortestCycle(G = (V, E)) :
    (initially the cycle length ℓ is ∞)
1   ℓ = ∞
    (consider every edge of G)
2   for each (v, w) ∈ E :
        (remove edge (v, w) from G)
3       G' = (V, E − {(v, w)})
4       Run BFS starting in v
        (is w still reachable from v?)
5       if w is in the same BFS-tree as v
            (find the shortest path, add 1)
6           let d = dist(v, w) + 1
            (if shorter than the best cycle so far)
7           if d < ℓ
                (save the length of the shortest cycle)
8               ℓ = d
9   return ℓ
```

Question 3 (Splay trees).

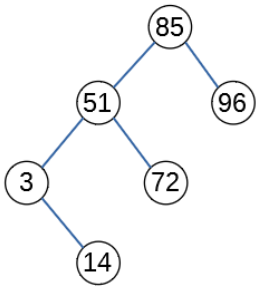

Figure 1: Intial Tree

(A) You have a Binary Search Tree in Figure 1. Insert two 2-digit numbers: $\overline{ab}$ and also $\overline{bc}$ as in a splay tree. (For example, if your Student ID ends with 789, then the numbers are 78 and 89).
For each insert show the following:

- How the tree looks after the insert, but before splaying.
- How the tree looks after the splaying (and how many zig-zig, zig-zag and/or zig operations you applied).

(B) Could the initial 6-node tree appear as a result of six successive node inserts (with splaying upon every insert)? If so, specify the sequence of six insert commands that would create the tree in Figure 1.

Question 4 (Graph Algorithm).

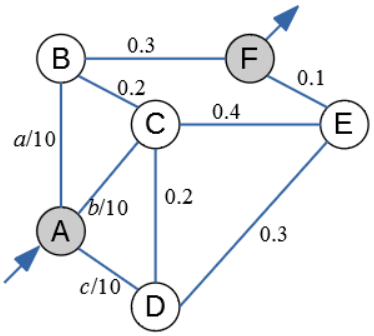

Figure 2: Paths in a Jungle

The graph with vertices A, B, C, D, E, F (Figure 2) shows the available paths in a jungle. Each edge on this graph displays the probability that a traveler will be bitten by a malaria mosquito and will be infected. Assume that mosquito bites are independent (the event of being bitten on one edge does not affect the probability of being bitten on a subsequent edge). Edges AB, AC, AD are marked with probabilities $a/10$, $b/10$ and $c/10$, where abc is your Student ID number.

(A) Find the *best path* from node A to node F such that the probability to be never bitten is the highest.

(B) Describe a method (as a pseudocode or a precise description) how to compute the best path in an arbitrary graph $G = (V, E)$ (given start and end vertices), if the probabilities to be bitten are known for every edge.

(C) Assume that you have Dijkstra's shortest path algorithm (available as a library function): It can find the shortest path between any vertices in a weighted undirected graph. You also have the graph $G = (V, E)$ for the "mosquito task" defined above with probabilities $p_{(v,w)} \in [0; 1]$ of being bitten for every edge $(v, w) \in E$.
Describe graph and edge weights that you would input into the Dijkstra's algorithm library function so that it outputs the best path for your "mosquito task". (You are allowed to modify the input graph itself and also assign any weights you want before passing them into the Dijkstra's algorithm.)