C. Diamond

time limit per test: 1 second
memory limit per test: 256 megabytes

- Implement a simple hierarchy of four related classes: `Person`, `Student`, `Employee`, and `TA`.
- Base abstract class (`Person`) contents:
  - Instance members: `fname` (first name) and `lname` (last name)
  - A class member: total personnel `count`, initially set to zero
  - A parameterized constructor: set `fname`, `lname`, and increments personnel count
  - A pure virtual function `string getEmail()`
  - A virtual default destructor and deleted copy constructor
- Two intermediate classes (`Student` and `Employee`):
  - Both inherit from `Person` class and call its parameterized constructor using initializer lists
  - Both override the `getEmail` function such that:
    - Students get emails in the format `fname[0].lname@students.org`
    - Employees get emails in the format `fname[0].lname@employees.org`
    - Note that all email addresses should be lowercase
- Derived (`TA`) class: inherits both `Student` and `Employee` (multiple inheritance)
  - TA should be counted as one person and get an employee email

Constraints

You **must** use this code template. Modifications to the given main function will lead to penalty:

```
#include <iostream>
#include <vector>
#include <type_traits>
#include <assert.h>

using namespace std;

/*
Your classes here
*/

int main() {
    static_assert(is_abstract_v<Person>);
    static_assert(is_base_of_v<Person, Employee>);
    static_assert(is_base_of_v<Person, Student>);
    static_assert(is_base_of_v<Person, TA>);
    static_assert(is_base_of_v<Employee, TA>);
    static_assert(is_base_of_v<Student, TA>);
    static_assert(is_polymorphic_v<Student>);
    static_assert(is_polymorphic_v<Employee>);
    static_assert(is_polymorphic_v<TA>);
    static_assert(!is_copy_constructible_v<Person>);
    static_assert(has_virtual_destructor_v<Person>);

    string fname, lname;

    cin >> fname >> lname;
    Student* s = new Student(fname, lname);
    assert(Person::count == 1);

    cin >> fname >> lname;
    Employee* e = new Employee(fname, lname);
    assert(Person::count == 2);

    cin >> fname >> lname;
    TA* t = new TA(fname, lname);
    assert(Person::count == 3);

    vector<Person*> people = {s, e, t};
    for(auto& p: people) {
        cout << p->getEmail() << endl;
        delete p;
    }
}
```

Input

Three lines of input:

- First line: Student's first and last name
- Second line: Employee's first and last name
- Third line: TA's first and last name

There is guarantee that input will be valid

Output

Three lines of output:

- Generated email addresses for Student, Employee, and TA respectively

Each input/output line ends with a single end-line `\n` character.

Explanation

- The TA class demonstrates multiple inheritance from both Student and Employee
- Virtual inheritance ensures Person base class is shared properly
- The static_assert statements validate the required class relationships
- Email generation follows specific formatting rules with lowercase conversion

Example

input	Скопировать
Alice Cooper Bob Dylan Charlie Puth	

output	Скопировать
a.cooper@students.org b.dylan@employees.org c.puth@employees.org	