Case Study -1

This problem is to get you familiar with virtual functions. Create three classes *Person, Professor* and *Student*. The class *Person*should have data members name and age. The classes *Professor* and *Student* should inherit from the class *Person*.

The class *Professor* should have two integer members: *publications* and cur_id. There will be two member functions: *getdata*and *putdata*. The function *getdata* should get the input from the user: the *name, age* and *publications* of the professor. The function *putdata* should print the *name, age, publications* and the cur_id of the professor.

The class *Student* should have two data members: *marks*, which is an array of size and cur_id. It has two member functions: *getdata* and *putdata*. The function *getdata* should get the input from the user: the *name, age*, and the *marks* of the student in subjects. The function *putdata* should print the *name, age*, *sum* of the marks and the cur_id of the student.

For each object being created of the *Professor* or the *Student* class, sequential id's should be assigned to them starting from .

Solve this problem using virtual functions, abstract methods, constructors and static class variables. You can create more data members if you want.

Note: Expand the main function to look at how the input is being handled.

Input Format

The first line of input contains the number of objects that are being created. If the first line of input for each object is , it means that the object being created is of the *Professor* class, you will have to input the *name, age* and *publications* of the professor.

If the first line of input for each object is , it means that the object is of the *Student* class, you will have to input the *name, age*and the *marks* of the student in subjects.

Constraints

$1 \leq len_{name} \leq$ name, where len_{name} is the length of the name.

$1 \leq age \leq 80$

$1 \leq publications \leq 1000$

$0 \leq marks \leq 100$, where marks is the marks of the student in each subject.

Output Format

There are two types of output depending on the object.

If the object is of type *Professor*, print the space separated *name*, *age*, *publications* and *id* on a new line.

If the object is of the *Student* class, print the space separated *name*, *age*, the *sum of the marks* in subjects and *id* on a new line.

Sample Input

```
4
1
Walter 56 99
2
Jesse 18 50 48 97 76 34 98
2
Pinkman 22 10 12 0 18 45 50
1
White 58 87
```

Sample Output

```
Walter 56 99 1
Jesse 18 403 1
Pinkman 22 135 2
White 58 87 2
```

Bonus Points

The Project can be built using Makefile.

Constraints are checked using exceptions.

Each class has its own source and header files.

Code can be compiled by GNU GCC>=4.8.4 on Ubuntu>14.04.