CENG315

Homework #2

(Due: November 5th, Friday 23:55)

You are the owner of a candy store and today's customer is a teacher. She wants to buy various kinds of candies for her students and to play a little game with you to entertain her students. So you put various candies in a bag. In the bag, there are K kinds of candies ($1 \leq K \leq 100$). In the game, the teacher randomly picks a candy from the bag without letting you see it, and she wants you to find out what kind of candy it is. You can ask questions of the type "Is it of the kind k_1 or k_2 or k_3... k_N?" (where $N \leq K$). You can get just a "yes" or "no" answer from the teacher. When you correctly find what kind of candy it is, the teacher gives the candy to one of her students. After that, she picks another candy, and the game goes on until the bag becomes empty. Since you put the candies into the bag, you know exactly how many candies of each kind there are in the bag. Your goal is to find the correct kinds of candies by minimizing the total number of questions you ask until the bag gets emptied so that the children can eat the candies more quickly.

The kinds of candies are represented with integers 1 through K, consecutively. There are at most 100 kinds of candies and 100 candies of each kind. Interaction of your program with the teacher will be through the functions called *query* and *guess*. The *query* function allows your program to ask whether the candy just picked is of the kind k_1 or k_2 or ... k_N. The *guess* function allows your program to assert that the candy just picked is of the kind k. **In a typical run, for each candy picked, your program will perform a sequence of calls to *query*, followed by one call to *guess* (If you know the kind without needing to query (for example; there is just one kind of candies in the bag), just call *guess*). The call to *guess* must never fail, that is your program must always be asserting the correct kind for the candy.**

In this homework, you'll be given an object file (main.o) which contains these interface functions; namely *query* and *guess*. Further, the object file contains the *main* function. Since the *main* function is already included in our object file, you must **not** have a function called *main*; instead you must have a function called *play* as the definition below.

```
/* This function answers the following question. Is the candy picked of the kind kind[0] or ... or kind[size-1]? If yes return 1, otherwise return 0. This function will be provided by us in the object file. But you should put the declaration of the function in your code. */
int query(int size, int* kind);

/* This function assert that the candy picked is of the kind k. Returns 1 if assertion is correct, 0 otherwise. This function will be provided by us in the object file. But you should put the declaration of the function in your code. (Note that the return value is provided to help you with debugging. Recall that during grading a return value of 0 causes your program immediately fail the test case.) */
int guess(int k);
```

```
/* The definition of the function should be in your hw2.cpp file. This function will be called from our main function only once. */
void play(void)
{
    ...
}
```

Input Specification

The first line of the input file contains the number of the kinds (K) of candies ($1 \leq K \leq 100$). Each of the following K lines contains the number of candies of the respective kind such that the $(k_N+1)^{th}$ line of the input file contains the number of candies of kind k_N. For example; in the below sample input, 5 is the total number of candies of kind 2. The input file is named hw2.inp.

Sample Input:

```
3
1
5
3
```

Output Specification

In this homework, you aren't going to generate an output file. To help you with debugging, in our main function, an output file including the kinds of candies that are picked (in order) and at the end the total number of questions you asked for one input file will be generated; namely hw2.out. If you guess a kind wrongly, the output file will contain -1 as the number of questions that are asked. You can either give a command line argument (an integer) to main or don't give an argument. If you don't give a command line argument, the order of the picked items can change in every call. If you give a number as a command line argument(for example; 30), the candies will still be picked randomly but the order of the picked items do not change in another call with that integer. This will help you with debugging.

Below, there is a possible output for the sample input. The first 9 lines are the kinds of the picked candies, consecutively and 11 is the total number of questions that are asked.

Sample Output:

```
2
3
3
2
2
1
2
3
2
11
```

Limits

Time limit for an input: 10 sec

Memory Limit: 32 mb

Grading

The grader will simply count the number of times your program calls *query* until there are no candies left in the bag. In case a call to *guess* fails to identify the kind of the candy picked, your program gets zero point for that test case (input). If the total number of *query* operations performed for an input is above our minimum, but less than 3/2 of this minimum, you still can get partial credit for that test case. If it is equal to or less than of our minimum then you'll get full credit for that input.

Specifications

- All the work should be done individually. Your homeworks will be checked for cheating. In case you cheat, you will get 0 from all your homeworks and disciplinary action will be taken.
- Submit a single file called "hw2.cpp" through the COW system.
- You can use the following command to compile your code:
  g++ hw2.cpp main.o -o hw2
- An autograder is available which will grade your homework with real inputs. The grade you get from autograder is final if there isn't any cheating issues.
- You can use the autograder up to 3 times per day.
- Late submission isn't allowed.