3. Get Hundred Points!

You are taking a test with 100 multiple-choice questions. Each question is numbered from 0 through 99. The answer to question i (for each $0 \leq i \leq 99$) is one of `A`, `B`, and `C`. The test is quite weird that it gives you the information that A questions have answer `A`, B questions have the answer `B`, and C questions have the answer `C`.

You may submit up to 100 answer sheets. Each time you submit an answer sheet, you will get informed on how many problems you got correct. However, there is a restriction: you should write A `A`s, B `B`s, and C `C`s in each answer sheet. You may use previous results to write a new answer sheet to submit.

Write a program that finds all the answers correctly by using an appropriate strategy.

Implementation details

You should implement the following function:

```
string GetHundredPoints(int A, int B, int C)
```

- A: the number of questions with answer `A`
- B: the number of questions with answer `B`
- C: the number of questions with answer `C`
- This function is called once for each test case.
- This function should return a string X of length 100. For each $0 \leq i \leq 99$, $X[i]$ should be the answer of question i (one of `A`, `B`, and `C`).

The function `GetHundredPoints` can call the following grader function:

```
int Mark(string S)
```

- S: your answer sheet. For each $0 \leq i \leq 99$, $S[i]$ should be your answer to question i (one of `A`, `B`, and `C`). The number of `A`s, `B`s, and `C`s in S should be A, B, and C, respectively.
- This function returns the number of correct answers in your answer sheet.
- This function can be called up to 100 times.

If some of the above conditions are not satisfied, your program is judged as `Wrong Answer`. Otherwise, your program is judged as `Accepted`.

Example

Suppose the answer of the text is as below (`ABCC` x 25):

```
ABCCABCCABCC...ABCC
```

The grader makes the following function call:

```
GetHundredPoints(25, 25, 50)
```

Let us consider the following calls to the function `Mark`:

- `Mark("CACBCACB...CACB")` (`CACB` x 25): The function returns 25.
- `Mark("ABAB...ABCC...CC")` (`AB` x 25, `C` x 50): The function returns 52.

The answer is `ABCCABCCABCC...ABCC`.

Constraints

- $0 \leq A, B, C \leq 100$
- $A + B + C = 100$

In this problem, the grader is NOT adaptive. This means the answers to all the problems are fixed at the beginning of the running of the grader and they do not depend on the queries asked by your solution.

Subtasks

1. (33 points) $C = 0$
2. (67 points) No additional constraints.

Sample grader

You can download the sample grader package on the same page you downloaded the problem statement. (scroll down if you don't see the attachment)

If you use IDEs like Visual Studio, Eclipse or Code::Blocks, then import `hundred.cpp`, `hundred.h` and `grader.cpp` into one project and you will be able to compile all these files at once.

If you want to compile by yourself, refer to the compilation commands in the statement page.

You should submit only `hundred.cpp`.

Input format

- line 1: X

Output format

If your program is judged as `Accepted`, the sample grader prints `Correct` in the first line and q in the second line, with q the number of calls to `Mark`.

If your program is judged as `Wrong Answer`, the sample grader prints the error message in the first line.