Kick Start 2019 - Practice Round

Analysis: Number Guessing

Test set 1

Since **A** = 0 and **B** = 30 in this test set, and since we get **N** = 30 tries per test case, we can simply guess every number from 1 to 30 until the judge sends back `CORRECT`.

Test set 2

In test set 2, since the answer could be anywhere in the range (0, 10^9] and we still have only 30 guesses, we will use binary search.

Initially, we know the answer P is in [1, 10^9], which is a big range! To cut that range by half, our first guess will be (1 + 10^9) / 2 = 5×10^8. If the judge sends back `TOO_SMALL`, we will know that P is in [1, 5×10^8). Similarly, if the judge sends back `TOO_BIG`, P is in (5×10^8, 10^9]. Otherwise, P is 5×10^8 and we are done.

We will cut that range further by making our next guess the middle number in that range. Again, based on the judge response that we get, we will know that either we have guessed P correctly, or P is in the upper or lower half of the range. We will do this repeatedly, until `CORRECT` is received.

Each time we make a wrong guess, the range that we must examine next will always be at most half the size of our previous range. So, it will take at most $\log_2 10^9 = 29.897353 < 30$ tries to guess P correctly.

Sample Solutions

This problem was intended as an opportunity to get used to our interactive judges. Here are some example solutions in all languages that we support so far:

Bash:

```
read t
for p in $(seq 1 $t); do
  read -a line
  a=${line[0]}
  b=${line[1]}
  read n
  head=$(( a+1 ))
  tail=$b
  while true; do
    mid=$(( (head+tail)/2 ))
    echo $mid
    read s
    if [[ "$s" == "CORRECT" ]]; then
      break
    elif [[ "$s" == "TOO_BIG" ]]; then
      tail=$(( mid - 1 ))
    elif [[ "$s" == "TOO_SMALL" ]]; then
```

```
      head=$(( mid + 1 ))
    else
      # Wrong answer; exit to receive Wrong Answer judgment
      exit 0
    fi
  done
done
```

C:

```
#include <stdio.h>
#include <string.h>

int main() {
  int T; scanf("%d", &T);

  for (int id = 1; id <= T; ++id) {
    int A, B, N, done = 0;
    scanf("%d %d %d", &A, &B, &N);
    for (++A; !done;) {
      int mid = A + B >> 1;
      char result[32];
      printf("%d\n", mid);
      fflush(stdout);
      scanf("%s", result);
      if (!strcmp(result, "CORRECT")) done = 1;
      else if (!strcmp(result, "TOO_SMALL")) A = mid + 1;
      else B = mid - 1;
    }
  }
  return 0;
}
```

C#:

```
using System;

public class Solution
{
  static public void Main ()
  {
    int num_test_cases = Convert.ToInt32(Console.ReadLine());
    for (int i = 0; i < num_test_cases; ++i) {
      string[] lo_hi_s = Console.ReadLine().Split(' ');
      int[] lo_hi = Array.ConvertAll(lo_hi_s, int.Parse);
      int num_tries = Convert.ToInt32(Console.ReadLine());
      int head = lo_hi[0] + 1, tail = lo_hi[1];
      while (true) {
        int m = (head + tail) / 2;
        Console.WriteLine (m);
        string s = Console.ReadLine();
        if (s == "CORRECT") break;
        if (s == "TOO_SMALL")
        {
```

```
          head = m + 1;
        }
        else
        {
          tail = m - 1;
        }
      }
    }
  }
}
```

C++:

```
#include <iostream>
#include <string>

int main() {
  int num_test_cases;
  std::cin >> num_test_cases;
  for (int i = 0; i < num_test_cases; ++i) {
    int lo, hi;
    std::cin >> lo >> hi;
    int num_tries;
    std::cin >> num_tries;
    int head = lo + 1, tail = hi;
    while (true) {
      int m = (head + tail) / 2;
      std::cout << m << std::endl;
      std::string s;
      std::cin >> s;
      if (s == "CORRECT") break;
      if (s == "TOO_SMALL")
        head = m + 1;
      else
        tail = m - 1;
    }
  }
  return 0;
}
```

Go:

```
package main

import (
  "fmt"
  "strings"
)

func main() {
  var t int
  fmt.Scanf("%d", &t)
  for i := 1; i <= t; i++ {
    var a, b, n int
```

```
    fmt.Scanf("%d %d", &a, &b)
    a = a + 1
    fmt.Scanf("%d", &n)
    for {
      m := (a + b) / 2
      fmt.Println(m)
      var str string
      fmt.Scanf("%s", &str)
      if strings.EqualFold(str, "CORRECT") {
        break
      } else if strings.EqualFold(str, "TOO_SMALL") {
        a = m + 1
      } else if strings.EqualFold(str, "TOO_BIG") {
        b = m - 1
      }
    }
  }
}
```

Haskell:

```
import System.IO

getNum :: IO Int
getNum = do
    x <- getLine
    let n = read x :: Int
    return n

bisect :: Int -> Int -> Int -> String -> IO ()
bisect a b m "CORRECT" = return ()
bisect a b m "TOO_SMALL" = singleCase (m+1) b
bisect a b m "TOO_BIG" = singleCase a (m-1)

query :: Int -> IO String
query m = do
    putStrLn ( show m )
    hFlush stdout
    x <- getLine
    return x

singleCase :: Int -> Int -> IO ()
singleCase a b = do
    let m = (a+b) `div` 2
    response <- query m
    bisect a b m response
    return ()

solve :: Int -> IO ()
solve 0 = return ()
solve n = do
    [a, b] <- fmap(map read.words)getLine
    _ <- getNum
    singleCase (a+1) b
    solve (n-1)
```

```
main = do
    hSetBuffering stdout NoBuffering
    t <- getNum
    solve t
```

Java:

```
import java.util.Scanner;

public class Solution {
  public static void solve(Scanner input, int a, int b) {
    int m = (a + b) / 2;
    System.out.println(m);
    String s = input.next();
    if (s.equals("CORRECT")) {
      return;
    } else if (s.equals("TOO_SMALL")) {
      solve(input, m + 1, b);
    } else {
      solve(input, a, m - 1);
    }
  }

  public static void main(String args[]) {
    Scanner input = new Scanner(System.in);
    int T = input.nextInt();
    for (int ks = 1; ks <= T; ks++) {
      int a = input.nextInt();
      int b = input.nextInt();
      int n = input.nextInt();
      solve(input, a + 1, b);
    }
  }
}
```

JavaScript:

```
var readline = require('readline');
var rl = readline.createInterface(process.stdin, process.stdout);

expect = 'begin';
rl.on('line', function(line) {
  if (expect === 'begin') {
    num_test_cases = parseInt(line);
    expect = 'lo_hi';
    case_counter = 0;
  } else if (expect === 'lo_hi') {
    lo_hi = line.split(' ');
    head = parseInt(lo_hi[0]) + 1;
    tail = parseInt(lo_hi[1]);
    expect = 'num_tries';
  } else if (expect === 'num_tries') {
    num_tries = line; // not used.
```

```
    expect = 'solve';
    mid = parseInt((head + tail) / 2);
    console.log(mid);
  } else if (expect === 'solve') {
    if (line === 'CORRECT') {
      ++case_counter === num_test_cases ? rl.close() : 0;
      expect = 'lo_hi';
    } else {
      line === 'TOO_SMALL' ? head = mid + 1 : tail = mid - 1;
      mid = parseInt((head + tail) / 2);
      console.log(mid);
    }
  }
}).on('close',function(){
    process.exit(0);
});
```

PHP:

```
<?php

function solve($a, $b) {
  $m = ($a + $b) / 2;
  printf("%d\n", $m);
  fscanf(STDIN, "%s", $s);
  if (strcmp($s, "CORRECT") == 0) {
    return;
  } else if (strcmp($s, "TOO_SMALL") == 0) {
    $a = $m + 1;
  } else {
    $b = $m - 1;
  }
  solve($a, $b);
}

fscanf(STDIN, "%d", $t);
for ($ks = 0; $ks < $t; $ks++) {
  fscanf(STDIN, "%d %d", $a, $b);
  fscanf(STDIN, "%d", $n);
  solve($a + 1, $b);
}
?>
```

Python2:

```
import sys

def solve(a, b):
  m = (a + b) / 2
  print m
  sys.stdout.flush()
  s = raw_input()
  if s == "CORRECT":
    return
```

```
  elif s == "TOO_SMALL":
    a = m + 1
  else:
    b = m - 1
  solve(a, b)

T = input()
for _ in xrange(T):
  a, b = map(int, raw_input().split())
  _ = input()
  solve(a + 1, b)
```

Python3:

```
import sys

def solve(a, b):
  m = (a + b) // 2
  print(m)
  sys.stdout.flush()
  s = input()
  if s == "CORRECT":
    return
  elif s == "TOO_SMALL":
    a = m + 1
  else:
    b = m - 1
  solve(a, b)

T = int(input())
for _ in range(T):
  a, b = map(int, input().split())
  _ = int(input())
  solve(a + 1, b)
```

Ruby:

```
$stdout.sync = true

def solve(a, b)
  m = (a + b) / 2
  puts m
  $stdout.flush
  s = STDIN.gets.chomp
  if s.eql? "CORRECT"
    return
  elsif s.eql? "TOO_SMALL"
    solve(m + 1, b)
  else
    solve(a, m - 1)
  end
end

t = STDIN.gets.chomp.to_i
```

```
ks = 1
while ks <= t
  a, b = STDIN.gets.split.map &:to_i;
  n = STDIN.gets.chomp.to_i
  solve(a + 1, b)
  ks = ks + 1
end
```