Introduction to Programming Competitions

Input/Output

Dr Vincent Gramoli | Lecturer
School of Information Technologies

THE UNIVERSITY OF SYDNEY

Automated judging

Automated judging

Manipulate test files

```
./a.out < input > myoutput
diff myoutput output
```

THE UNIVERSITY OF SYDNEY

Input

Input

You have several choices when it comes to reading text input:

- You can repeatedly get single characters from the input stream to process them one at a time:
  - getchar() in C
- You can repeatedly get formatted tokens to process them as needed
  - scanf() in C
  - Scanner in Java
- You can read the entire line as a single string and then parse its substrings and characters
  - gets() in C
- You can use modern input/output primitives such as streams if your language supports them (for string, characters of something else).
  - BufferedReader in Java

Input

getchar() reads the next character

```
while (c = getchar()) != '\n') {
  if (c == EOF) return;
  if (c != ' ') {
    value = c;
  }
}
```

Input

C++ example to read from the input file

- An input file represents test cases with at most 20 sets of 20 integers
- 1st line indicates the number of test cases:
  - Each test case starts with M, some capacity integer, and C, # of sets that follows
  - For each set, a line starts with K (its size) and the integers it contains

```
int M, C, integers[25][25]; // choose larger bounds
int main() {
    int i, j, TC;
    scanf("%d", &TC); // store test cases
    while (TC--) {
      scanf("%d %d", &M, &C);
      for (i = 0; i < C; i++) {
        scanf("%d", &integers[i][0]; j++) { // get size
        for (j = 1; j <= integers[i][0]; j++)
            scanf("%d", integers[i][j]);    // get integers
    }}
```

C/C++ example

THE UNIVERSITY OF SYDNEY

Non-standard input

Take a problem with a non-standard input format:

- The first line of input is an integer N
- This is followed by N lines, each starting with the character ‘0’
- Followed by a dot ‘.’
- Followed by a up to 100 digits
- Terminated by three dots ‘…’

```
3
0.1227...
0.517611738
0.7341231223444344389923899277...
```

Non-standard input

C/C++ example

```
#include <cstdio>
using namespace std;

int N;          // using global variables
char x[110];    // array size a bit larger than needed

int main() {
  scanf("%d\n", &N);
  while (N--) {  // we simply loop from N, N-1, N-2, ..., 0
    scanf("0.%[0-9]...\n", &x);   // '&' optional as x char array
    printf("the digits are 0.%s\n", x);
  }
} // return 0;
```

Java Example

Example

Hardwood species problem (Uva#10226)

› Scanner example of input (Java)

```
public void solve() throws IOException {
  Scanner in = new Scanner(System.in);
  while(in.hasNextLine()) record(in.nextLine());
    print();
}
```

Example

Hardwood species problem (Uva#10226)

› Scanner example of input (Java)

```
public void solve() throws IOException {
  Scanner in = new Scanner(System.in);
  while(in.hasNextLine()) record(in.nextLine());
    print();
}
```

This is too slow!

Example

Hardwood species problem (Uva#10226)

› Buffered input in Java

```
public void solve() throws IOException {
  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
  String s;
  while((s = br.readLine()) != null) record(s);
    print();
}
```

Example

Hardwood species problem (Uva#10226)

› Constraints were 5 second running time / 32M of memory max

Solutions	Time	Memory	Results
Scanner	14s	6K	TLE
Buffered	3.5s	6K	AC

(Values observed while submitting on PKO online judge)

Example

- Problem submissions must run under a certain time limit
  - 5 seconds?
  - TLE error: Time Limit Exceeded
- Input parsing can be time consuming!
- Be careful about input parsing in Java
  - Scanners can be useful but
  - BufferedReader can be much more efficient

Output

Input

You have several choices when it comes to writing to the standard output:

- You can write a series of different types formatted appropriately
  - printf() in C
- You can write pretty a string or another type using
  - puts() in C
- You can write character by character, not really used in practice though
  - putchar() in C
- You can use modern input/output primitives such as streams if your language supports them (for string, characters of something else).

Output

Leverage C features when coding C++

- Not many C/C++ pogrammers are aware of partial regex capabilities built into the C standard I/O library
- Although scanf/printf are C-style I/O routines, they can still be used in C++ code
- Many C++ programmers force themselves to use cin/cout all the time even though it is sometimes not as flexible as scanf/printf and is also far slower

Output

Java

- Buffering executes faster than scanning
- Use the BufferedReader/BufferedWriter classes as follows:

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
// Note: String splitting and/or input parsing is needed afterwards
```

```
PrintWriter pr = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(System.out)));
// PrintWriter allows us to use the pr.printf() function
// do not forget to call pr.close() before exiting your Java program
```

THE UNIVERSITY OF SYDNEY

Output

Format the output with printf: %[flags][width][.precision][length]specifier

- Specifier:
  - d integer representation, u unsigned octal
  - c character, s string of characters
  - f floating point representation, a hexadecimal floating point representation
  - e scientific notation
- Precision
  - For integer representations, this is the min of digits to print (*:unspecified)
  - For others, this is the number of digits after the decimal point (*:unspecified)
- Flag and width
  - 0 left-pads the number with 0 (width=num) or blank space (width=*)
  - + forces sign before number, - left-justifies number

Example

- Hardwood species problem (Uva#10226)

```
public void print() {
  for (Map.Entry<String, Integer> entry : tm.entrySet()) {
    System.out.printf("%s %.4f\n", // 4 digits after decimal point
        entry.getKey(),
        (entry.getValue())*100.0/total);
  }
}
```

Test

Testing

› Check your code output with ‘diff’ / ‘vimdiff’ (Unix-based) or ‘fc’ (Windows).

```
g++ a.cpp

./a.out < input > myoutput
diff myoutput output
```

Testing

- Write scripts to compile and test in debug mode,

```
g++ -O0 -c A.cpp A && /.A < 1.in > 1.out
```

- in non-debug mode…

```
g++ -O2 -c A.cpp A && /.A < 1.in > 1.out
```

Test

Remarks:

- Introduce multiple identical test cases consecutively in the same run for problems with multiple test cases in a single run.
- Include corner cases: N=0, N=1, negative values, large values that do not fit 32-bit signed integer, etc.
- Very easy to get a Presentation Error
  - Be careful at white space
  - Extra ‘0’

I/O

Problem: given two integers in one line, output their sum in one line

A. The number of test-cases is given in the first line of the input

A.cpp

```
int TC, a, b;
scanf("%d", &TC); // number of test cases
while (TC--) {  // shortcut to repeat
  scanf("%d %d", &a, &b); // compute answer
  printf("%d\n", a + b); // on the fly
}
```

A.in

```
3
1 2
5 7
6 3
```

g++ A.cpp –c A; ./A <A.in >A.out

A.out

```
3
12
9
```

Problem: given two integers in one line, output their sum in one line

B. The multiple test-cases are terminated by special values

B.cpp

```
int a, b;
// stop when both integers are 0
while (scanf("%d %d", &a, &b), (a||b)) {
  printf("%d\n", a + b);
}
```

B.in

```
1 2
5 7
6 3
0 0
```

g++ B.cpp –c B; ./B <B.in >B.out

B.out

```
3
12
9
```

Problem: given two integers in one line, output their sum in one line

C. The multiple test-cases are terminated by EOF (end of file)

C.cpp

```
int a, b;
// scanf returns the number of items read
while (scanf("%d %d", &a, &b) == 2) {
  printf("%d\n", a + b);
}
```

C.in

```
1 2
5 7
6 3
```

g++ C.cpp –c C; ./C <C.in >C.out

C.out

```
3
12
9
```

Problem: given two integers in one line, output their sum in one line

C. The multiple test-cases are terminated by EOF (end of file)

C2.cpp

```
int a, b;
// scanf returns the number of items read
while (scanf("%d %d", &a, &b) != EOF) {
  printf("%d\n", a + b);
}
```

C.in

```
1 2
5 7
6 3
```

g++ C2.cpp –c C2; ./C2 <C.in >C.out

C.out

```
3
12
9
```

Shortcuts

Shortcuts

Headers (C++)

contest.h:

```
typedef long long ll;
typedef pair<int, int> ii;
typedef vector<ii> vii;
typedef vector<int> vi;
#define INF 1000000000 // 1 billion safer than 2B for Floyd Warshall's
#define REP(i, a, b) for (int i = (int) a; i < (int) b; i++)
```

One can add useful libraries: stdio, cmath, cstring, etc.

All solutions simply have to start with:

```
include<contest.h>
...
```

Expressions

```
ans = a ? b : c;  // if (a) ans = b; else ans = c
ans += val; // ans = ans + val
index = (index + 1) % n; // index++; if (index >= n) index = 0;
index = (index + n - 1) % n; // index--; if (index < 0) index = n-1
int ans = (int)((double)d + 0.5) // rounding to nearest integer
```

Shortcuts

Minimum/maximum (C++)

```
ans = (ans < new_computation) ? ans : new_computation;
```

```
#include<algorithm>
...
ans = std::min(ans, new_computation)
```

```
#include<sys/param>  // debian / FreeBSD
...
ans = MIN(ans, new_computation)
```

Shortcuts

Code factorization (Java)

```
void p(String f, Object...params) {  // signature is very short
    System.out.printf(f, param);
}
...
p("DEBUG: " + z + "\n");
```

```
void d(Object z) { // signature is very short
  if (debug) {
    if (z instanceof Object[]) {
      // string representation of the deep contents of z
      p("DEBUG: " + Arrays.deepToString((Object[]) z) + "\n");
    } else {
      p("DEBUG: " + z + "\n");
    }
  }
}
```

Problem with I/O

THE UNIVERSITY OF SYDNEY

ANZAC 2012 Round 3

Problem

Problem E — Enlarged Art

Your grandfather recently developed a passion for the internet and new technologies. He is now discovering the wonders of ASCII art. Unfortunately, his sight is not as it once was, and he has difficulty seeing the pictures.

Moreover, he cannot merely increase the font size as he usually does, because doing so makes the characters stand out more on their own and less as a whole. You come up with the following solution to write a program which duplicates the ASCII lines in order to make them look **bold**.

Input

The first line of the input will contain n, the number of test cases. For each test case, the first line will contain the positive integers w and h, both of which will be at most 100. Then h lines will follow, each containing w ASCII characters. The character '.' (dot) will denote the background ASCII "color", and the only other foreground color will be '*' (star).

Output

For each image, add a 1-character wide border of dots around the image, and proceed to replace every single star character in the original picture with a diamond,

```
 *
***
 *
```

where the middle star is positioned where the original star was. In the output, separate each test case with "---".

Sample Input	Sample Output
2 2 1 *. 3 3 .*. *.* .*.	.*.. ***. .*.. --- ..*.. .***. ***** .***. ..*..