Contest strategies

We have three people, one computer, five hours and up to twelve problems. What to do?

Roles

- Coder.
- Mathematician (aka thinker).
- Tester.

Note, that role of one person could change from time to time.

Note, that a person could carry more than one role.

Team strategies examples.

- C + M + T. Every person does the thing he does best. But the coder could be exhausted during first three or four hours.
- C + C + M. Because testing process is very important, every team member should be a bit tester in this case.
- S + S + S. (Three Stars). They can loose to weaker teams because of lack of teamwork.

The start of the contest

- Usually, all teams do almost the same during the start.
- One person sets up the environment.
- Two other read problems, one from the first, one from the last.
- If a third person manages with set up before simple problem is found, he joins the rest and reads problems from the middle.
- When the environment is set up and simple problem is found, the most suitable member writes a solution for it.
- Up to the end of first hour every person should know the statement of every problem except maybe problems, which are already accepted by the team.

Ideas or how to solve a problem?

- Ideas testing: if you think you have a solution, give your idea a check on sample cases and some small test cases.
- Tell your ideas to teammate, he can point you to wrong parts.
- If you have no idea, discussing with teammate can speedup the thinking process dramatically.

Before writing the code

- When planning the code keep in mind, that you will need to debug it later.
- It’s good to split up your code into small procedures and functions – they are easy to plan afterwards.
- Before writing, think about all needed data structures and algorithms, make sure you know, what and how you will implement.
- Think about all technical details of your code, make sure you will not face some unexpected implementation problem during writing the code.
- If you have any formulas (mathematical, geometric or dynamic), write down all of them on a piece of paper. Carefully check the indices if any.

Writing the code

- Writing top-down: you write the main function (calling some procedures maybe), then needed procedures, then procedures needed for first procedures and so on.
- Writing bottom-up: you start with small procedures and then go to top until you reach the main function.

Testing your code

- Testing is a crucial process: without it you can hardly accept any problem except maybe “A+B”.
- Test you program on a hand-made tests. They should include minimal and other corner cases.
- Test your program on the sample test cases. Sometimes they contain a good testset.
- Sometimes you can have a “beautiful” test – do not hesitate to use it. These can be some patterns, for example string “abacaba” etc

Testing your code (continued)

- Maximal tests are important: you can catch TL, ML, RE using them in most cases.
- To make a maximal test you often need a generator – you will not spend much time to do it.
- Make assertions in your program, they can help you to check if something is wrong.

Stress-testing

- Stress-testing is a technique used to run your program on a plenty of tests in small time.
- Usually you need the solution you want to check, a correct solution (it can be slow, or memory-consuming), a generator and a verifying program.

How to stress-test?

Consider following .bat – script:

```
@echo off
:loop
   gen >input.txt
   if errorlevel 1 goto exit
   del output.txt
   sol1
   if errorlevel 1 goto exit
   move output.txt output.ans
   sol2 if errorlevel 1 goto exit
   fc output.txt output.ans
   if errorlevel 1 goto exit
   goto loop
:exit
```

How to stress-test?

Or following .sh – script:

```
#!/bin/sh
while true; do
   ./gen >input.txt || break
   rm output.txt
   ./sol1 || break
   mv output.{txt,ans}
   ./sol2 || break
   export x=$((x + 1))
   echo $x
   diff output.{txt,ans} || break
done
```

How to write a generator?

- Use a random number generator.
- Keep in mind, that standard number generator produces uniformly distributed numbers in segment [0..RAND_MAX], where RAND_MAX is defined as 65536 in most compilers.
- You can initialize generator to produce different tests by calling **srand(time(NULL))**.
- But it is bad! Why? Because time() updates once a second, so you can get at most one test per second. The solution is to use **rdtsc** or **GetTickCount()**.

How to write a generator (continued).

```
long long Time()
{
    #ifdef __GNUC__
    long long res;
    asm volatile ("rdtsc" : "=A" (res));
    return res;
    #else
    int low, hi;
    __asm{
        rdtsc
        mov low, eax
        mov hi, edx
    }
    return (((long long)hi) << 32LL) | low;
    #endif
}
```

Code reading

- It is very useful to print your program and check its paper version.
- It is even more useful, if you are telling what is done in your program to your teammate.

Parallelism

- You have three persons and only one computer, thus you have 15 person-hours and only 5 computer-hours. So you are to spend computer time wisely.
- The computer should not stand still, at any time there should be something to do on it: setting up, writing code, debugging or testing.
- Do not spend much computer time to debugging, use code reading and paper instead.
- When someone is coding, two others can invent a solution for other problem, or find a bug in another solution.
- But for hard problems, do not leave coding for just one person – he could make lots of mistakes there, let one another watch him and third person to work on another problem (inventing solution or tests, reading code and so on).

Parallelism

- For very hard problems, work in three for some steps of solving (e.g. inventing solution or testing).
- To save some computer time, write parts of the code on paper (while other person is working on computer).
- If you are stuck in something (technical detail, debugging or testing), ask your teammates for help.

Some unexpected bugs

Do not divide by zero

```
var a, b : integer;
begin
  read (a, b);
  writeln (a div b);
end.
```

Which values of a and b lead to “Division by zero” error?

Do not divide by zero

1. Only b = 0
2. b = 0 and one more value of b
3. b = 0 and one more pair (a, b)
4. None of the above

Do not divide by zero

- Correct answer: b = 0 and
  (a = -2147483648, b = -1)

Powers of two

```
#include <cstdio>
#include <cassert>

int main () {
  int s = 0, t;

  scanf ("%d", &t);
  assert (! (t & (t - 1)));
  while (t) {
    t >>= 1;
    s += t;
  }
  printf ("%d\n", s);
  return 0;
}
```

Powers of two

Will this program hang?

1. Always will hang
2. Will hang only for one value of t
3. Always will terminate
4. None of the above

Powers of two

- Correct answer: will hang for t = -2^31 (-2147483648)
- In java you can cope with it using logical shift (>>>), in C use unsigned types.

Type cast in java

Program 1:

```
import java.util.*;

public class t1 {
  public static void main (String args [])
  throws Exception {
    Scanner in = new Scanner (System.in);
    int a = in.nextInt ();
    long b = in.nextLong ();
    a = a + b;
    System.out.println (a);
  }
}
```

Type cast in java

Program 2:

```
import java.util.*;

public class t2 {
  public static void main (String args [])
  throws Exception {
    Scanner in = new Scanner (System.in);
    int a = in.nextInt ();
    long b = in.nextLong ();
    a += b;
    System.out.println (a);
  }
}
```

Type cast in java

Do these programs compile?

1. Both compile
2. Both don’t compile
3. Only first compiles
4. Only second compiles

Type cast in java

Correct answer:

There is strict type cast in java, so first program does not compile (you can not put **long** into **int**). But second compiles!

Using of functions

```
#include <cstdio>
#include <cassert>

int nextInt () {
  int tmp;
  assert (scanf ("%d", &tmp) == 1);
  return tmp;
}

void display (int a, int b) {
  printf ("%d %d\n", a, b);
}

int main () {
  display (nextInt (), nextInt ());
  return 0;
}
```

Using of functions

What will it output having “2 3” in input?

1. 2 2
2. 3 2
3. 2 3
4. Result is undefined…

Using of functions

Correct answer: result is undefined.

Order of calculating of parameters is not fixed in C/C++.