

# Competitive Coding

*An approach to analyzing and problem solving*

# Questions on C++



# Understanding the question

# EG: Multiply a pair of ints

- First line of input: T # of test cases
- The following T lines: 2 ints separated by space in format:
  - A B
  - where A, B are ints, and  $-2^{20} \leq A, B \leq 2^{20}$
  - Print the result
  - What are some problems you might have?



- Sample input:

- 3

- 5 2

- 200 -3

- 9 7

- Sample output:

- 10

- -600

- 63

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

int main() {
    int T;
    cin >> T;

    for (int t = 0; t < T; t++) {

        int A, B;
        cin >> A >> B;

        cout << A * B << endl;
    }

    return 0;
}
```

Wait.. what if we had

$$A = B = 2^{20}?$$



We get 0 ( ! )

Why?



# 32-bit int

- Integers are stored in 32 bits (and are signed)
- Largest it can store is  $2^{31}-1$
- Larger than max is called overflow

Solution: use  
long  
or  
long long



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

int main() {
    int T;
    cin >> T;

    for (int t = 0; t < T; t++) {

        long long A, B;
        cin >> A >> B;

        cout << A * B << endl;
    }

    return 0;
}
```

# Practice Questions!



# Q1: Return the largest number

- First line of input: T # of test cases
- The next T test cases have input format:
  - N A B ....
  - Where N is the number of integers passed, A B ... are integers;  $-2^{20} \leq A, B, \dots \leq 2^{20}$
- Print the output

## Q2: Find the number of matches of a character in a string

- First input is the string to count from, no space
- second input is a char
- Count the number of the character in the string
- print the result
- no libraries except iostream is allowed



# Q3: Partition sizes

- First line is in the format such that:
  - A B
  - where A is the total size, B is the number of possible dividers
- second line is in the format such that:
  - there are B integers describing the distance from 0; each of these integers represent a divider
- Print all possible sizes of partitions
- Sample in out:
  - 6 2
  - 2 5
  - out: 1 2 3 4 5 6 (can you understand why?)

# Pointers



Imagine a guy pointing at a  
room, where the integer lives  
in

This guy is the pointer. He is  
pointing to the memory  
address of the lovely Integer



Declaring a pointer:

```
int *a;
```

declaring a variable:

```
int a;
```

\*a means dereference a  
&a means address of a



Yes, that's how we declare it. why?

```
int *a;
```

“dereference a to get the integer”

therefore a is a pointer

```
// pointer to a integer
int *a;
// an actual integer
int b = 1;

// assign address of b for a to point to
a = &b;
```

```
~
~
~
~
~
~
~
~
~
~
```



The true identity of  
arrays...

POINTER TO THE  
FIRST INTEGER!



```
int a[5];
```

```
// what does this line do?
```

```
int *b = &a[0];
```

```
// *b++ == a[1];
```

```
// *b++ == a[2];
```

```
// etc
```

```
// but be careful! It won't stop you from going to the 6th  
// element. What's the 6th element? We don't know, something  
// on memory. UNSAFE! DON'T DO IT!
```

```
□
```

```
~
```

```
#include <iostream>
int main(void)
{    //for testing purposes we have already defined an 4x4 matrix
    int myArray[4][4] = { {1,2,3,4}, {5,6,2,8},{4,6,7,3},{7,3,4,8} };
    int width = 4, height = 4;
    for (int i = 0; i < height; ++i)
    {
        for (int j = 0; j < width; ++j)
        {
            std::cout << myArray[i][j] << ' ';
        }
        std::cout << std::endl;
    }
}
```

[13:06:26] wifihost-108-138:Downloads Conanap\$



# Returning 2D arrays

```
int** bruh() {  
    // I see you  
    // this doesn't work you dummy  
    // you thought i'd teach you useless  
    // stuff?  
    return a[4][4];  
}
```



```
#include <iostream>
using namespace std;
int main() {
    extern int**bruh();
    bruh();
    return 0;
}
int** bruh() {
    return new int*[4];
}
```

Even when declared `int **a`,  
you can use  
`a[x][y]`



This code is trying to swap the contents of two variables. Why isn't it working?

```
void swap(int &a, int b) {  
  
    int tmp = a;  
  
    a = b;  
  
    b = tmp;  
  
}
```

This won't compile. Why?

```
double a = 10;
```

```
int *aptr = &a;
```



Which of the following gives the memory address of a variable pointed to by pointer a?

A. a;

B. \*a;

C. &a;

D. address(a);

What's the output?

```
int size = 3;
```

```
int arr[12] = {7, 2, 4};
```

```
char name[12] = "Jeff Jones";
```

```
int *iptr;
```

```
char *cptr;
```

```
iptr = arr;
```

```
cout << iptr[1] << " and " << *iptr << endl;
```

```
cptr = &name[5];
```

```
cout << cptr << endl;
```

```
cout << *(cptr + 3) << " and " << *(cptr-4) << endl;
```

```
cptr ++;
```

```
cout << name[3] << " and " << cptr[3] << endl;
```



What do all of the following do?

\*itr++;

\*++itr;

(\*itr)++;

# Fun question

Print out a 2D Array in the form of a matrix.

`Int[4][4] = { {1,2,3,4}, {5,6,2,8},{4,6,7,3},{7,3,4,8} };`

`|x x x x|`

`|x x x x|`

`|x x x x|`

`|x x x x|`, x in set of real numbers