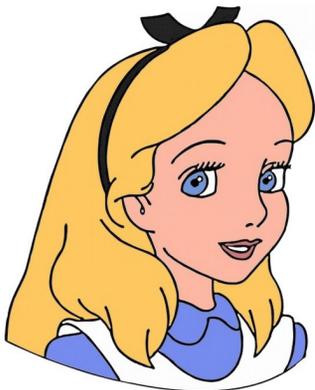


A Sample Exercise

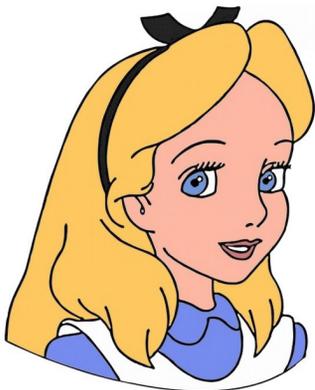
- Alice and Bob are fighting over who gets the last slice of pizza.
- They decide to settle the dispute by playing $n = 2k + 1$, $k \in \mathbb{N}$ rounds of Heads or Tails.
- Alice (resp. Bob) wins if the majority of the coin flips land on Heads (resp. Tails).
- Design an algorithm that, given n and the outcomes of the coin flips, decides who gets the last pizza slice.



A Sample Exercise

Input. The input consists of T instances, or *test cases* of the previous problem. The first line of the input contains the integer T . Each of the following lines represents a test case and consists of the number n and of a string s of n characters, where the i -th character of s is H if the i -th coin landed on heads and T otherwise.

Output. The output consists of T lines, one per test case, each containing a single character. In particular the i -th line should be “A” if alice won the i -th instance, and “B” otherwise.



A Sample Exercise

Example

Input: example.in

```
3
1 H
5 HHTHT
3 TTH
```

Output: example.out

```
A
A
B
```

Notes

A reasonable implementation should not require more than 1 second for each input file.

A Possible Solution

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

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

    while(T--)
        solve_testcase();

    return EXIT_SUCCESS;
}
```

A Possible Solution

```
void solve_testcase()
{
    int n;
    std::string s;
    std::cin >> n >> s;

    int number_of_H = 0;
    for(const char c : s)
        if(c == 'H')
            number_of_H++;

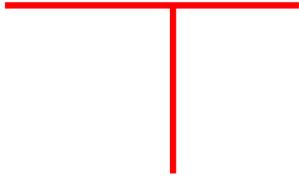
    std::cout << ((number_of_H > n/2) ? "A" : "B") << "\n";
}
```

Compiling

```
g++ -std=c++20 -Wall -pedantic -O3 solution.cpp -o solution
```

Compiling

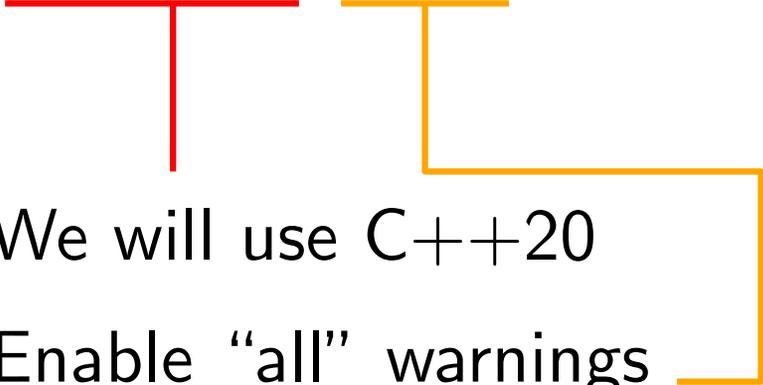
```
g++ -std=c++20 -Wall -pedantic -O3 solution.cpp -o solution
```



- We will use C++20

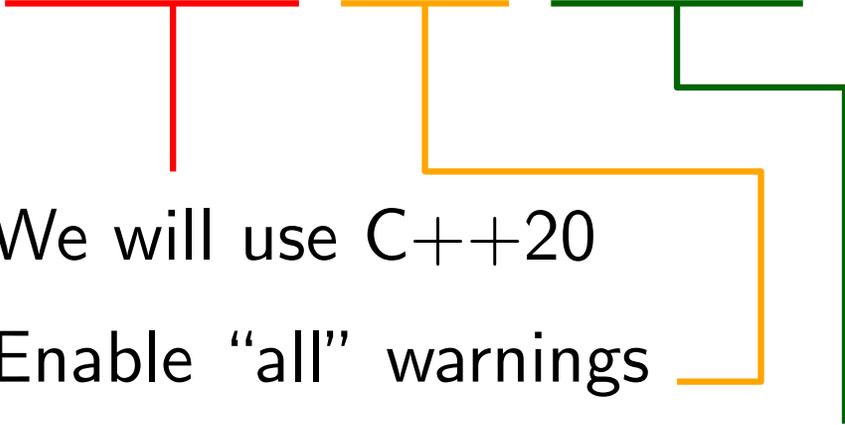
Compiling

```
g++ -std=c++20 -Wall -pedantic -O3 solution.cpp -o solution
```

- 
- We will use C++20
 - Enable “all” warnings

Compiling

```
g++ -std=c++20 -Wall -pedantic -O3 solution.cpp -o solution
```

- 
- We will use C++20
 - Enable “all” warnings
 - Strict compliance to the standard

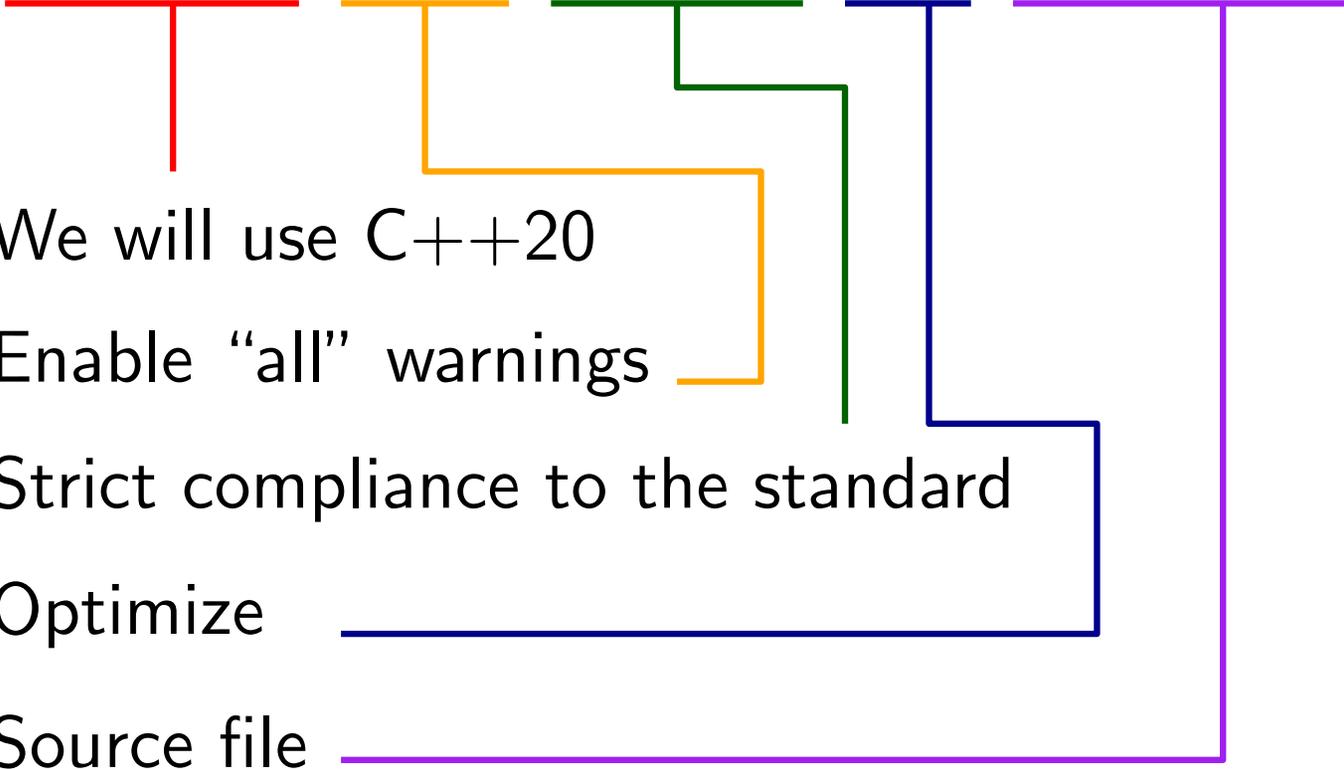
Compiling

```
g++ -std=c++20 -Wall -pedantic -O3 solution.cpp -o solution
```

- We will use C++20
 - Enable “all” warnings
 - Strict compliance to the standard
 - Optimize
-

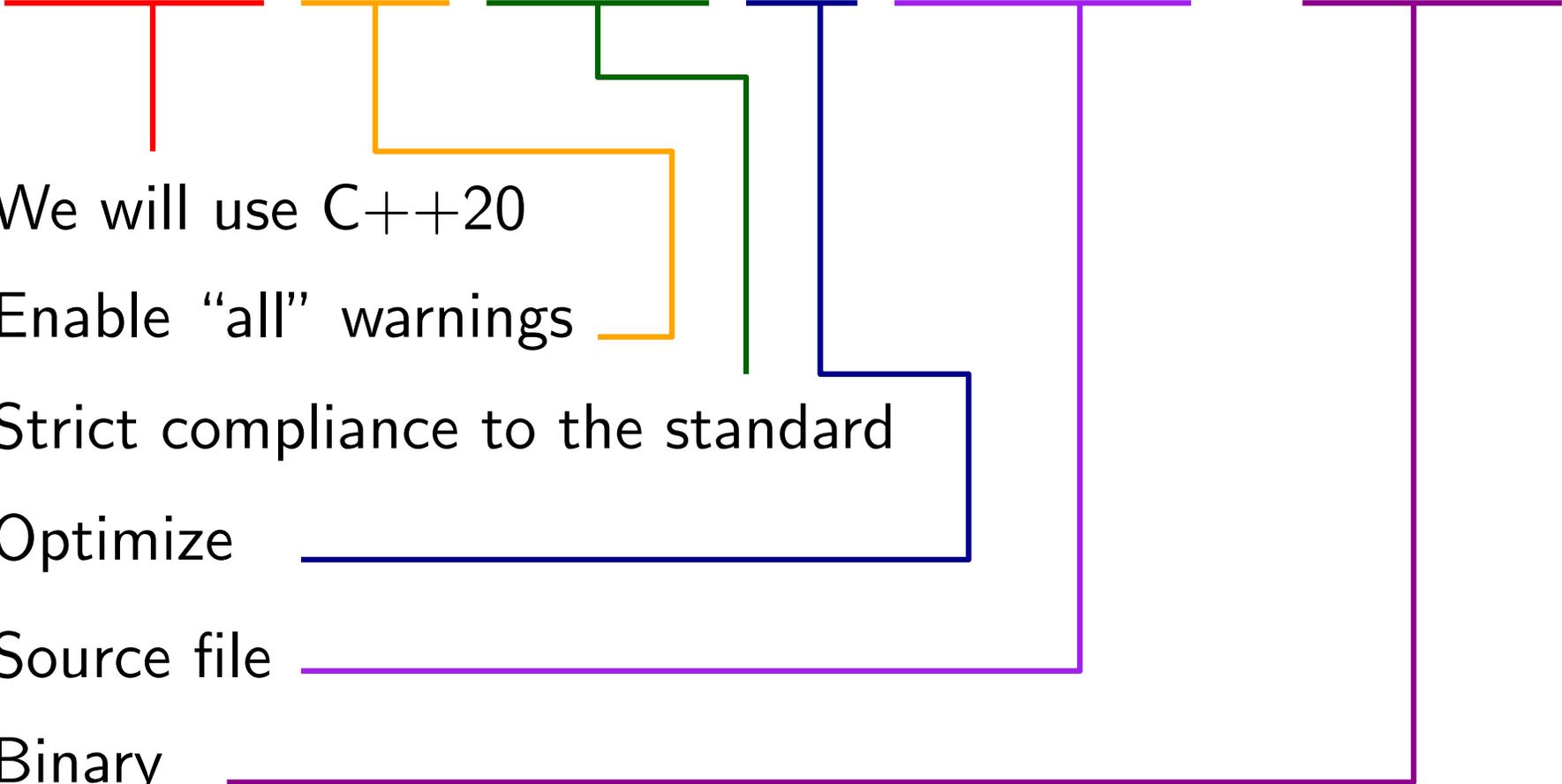
Compiling

```
g++ -std=c++20 -Wall -pedantic -O3 solution.cpp -o solution
```

- We will use C++20
 - Enable “all” warnings
 - Strict compliance to the standard
 - Optimize
 - Source file
- 
- A diagram consisting of five horizontal lines at the top, each a different color: red, orange, green, blue, and purple. From each line, a vertical line descends, then a horizontal line extends to the right, and finally a vertical line descends to connect to a bullet point in a list below. The red line connects to 'We will use C++20'. The orange line connects to 'Enable "all" warnings'. The green line connects to 'Strict compliance to the standard'. The blue line connects to 'Optimize'. The purple line connects to 'Source file'.

Compiling

```
g++ -std=c++20 -Wall -pedantic -O3 solution.cpp -o solution
```

- We will use C++20
 - Enable “all” warnings
 - Strict compliance to the standard
 - Optimize
 - Source file
 - Binary
- 
- A diagram consisting of several horizontal lines at the top, each a different color (red, orange, green, blue, purple, purple). From these lines, vertical lines descend and then turn horizontally to point to specific items in the list below. The red line points to 'We will use C++20'. The orange line points to 'Enable "all" warnings'. The green line points to 'Strict compliance to the standard'. The blue line points to 'Optimize'. The purple line on the left points to 'Source file'. The purple line on the right points to 'Binary'.

Checking the solution

```
$ ./solution < example.in > solution.out
```

Checking the solution

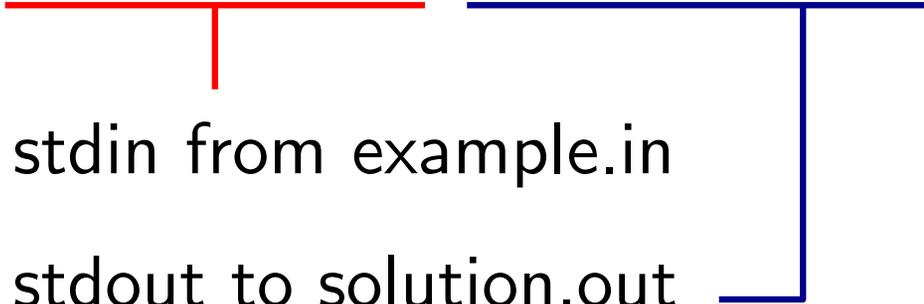
```
$ ./solution < example.in > solution.out
```



- Redirect stdin from example.in

Checking the solution

```
$ ./solution < example.in > solution.out
```

- Redirect stdin from example.in
 - Redirect stdout to solution.out
- 
- A diagram consisting of a red horizontal line above the first bullet point, a blue horizontal line above the second bullet point, a vertical red line connecting the red line to the first bullet point, and a blue L-shaped line connecting the blue line to the second bullet point.

Checking the solution

```
$ ./solution < example.in > solution.out
```

- Redirect stdin from example.in
- Redirect stdout to solution.out

```
$ diff -bBZ solution.out example.expected
```

```
$
```

- Ignore white space

Checking the solution

```
$ ./solution < example.in > solution.out
```

- Redirect stdin from example.in
- Redirect stdout to solution.out

```
$ diff -bBZ solution.out example.expected
```

```
$
```

- Ignore white space
- No output = files are identical.

Timing your solution

```
$ time ./solution < example.in > solution.out
```

```
real    0m0.005s
```

```
user    0m0.000s
```

```
sys     0m0.005s
```

Timing your solution

```
$ time ./solution < example.in > solution.out
```

```
real    0m0.005s
```

```
user    0m0.000s
```

```
sys     0m0.005s
```

Everything in a single command

```
$ (time ./solution < example.in) | diff -bBZ - example.expected
```

Tips

- Use the `assert()` macro in the `cassert` header.
- Test the assumptions you are making in your program.
- Perform sanity checks of your intermediate results.

```
#include<cassert>

void solve_testcase()
{
    int n;
    std::string s;
    std::cin >> n >> s;
    assert(s.size() == n);
}

[...]
```

Tips

- Use the `assert()` macro in the `cassert` header.
- Test the assumptions you are making in your program.
- Perform sanity checks of your intermediate results.
- Beware: checking assertions can take time.
- Disable assertions by defining the `NDEBUG` macro.

```
g++ -std=c++20 -Wall -pedantic -O3 -DNDEBUG solution.cpp -o solution
```

Tips

- Use the `assert()` macro in the `cassert` header.
- Test the assumptions you are making in your program.
- Perform sanity checks of your intermediate results.
- Beware: checking assertions can take time.
- Disable assertions by defining the `NDEBUG` macro.

```
g++ -std=c++20 -Wall -pedantic -O3 -DNDEBUG solution.cpp -o solution
```

- If your program requires heavy I/O, this might help

```
std::ios_base::sync_with_stdio(false);
```