accu 2022

LEARNING FROM C++ JUNIORS' MISTAKES

A CASE STUDY OF CODING BY EXAMPLE

AMIR KIRSH

About me

Lecturer
Academic College of Tel-Aviv-Yaffo and Tel-Aviv University

Developer Advocate at

INCREDIBUILD

Co-Organizer of the **CoreCpp** conference and meetup group

A credit note

This talk is based on a joint work with Dr. Iris Gaber

Goal

Analyzing actual mistakes of juniors

Goal

Analyzing actual mistakes of juniors

In an environment revealing specifically the “surprising mistakes”

Goal

Analyzing actual mistakes of juniors

In an environment revealing specifically the “surprising mistakes”

- ➢ done even though *the developer should have known better*
  - ❑ we just talked about that in class
  - ❑ you got an example of how to do it right

Goal

Analyzing actual mistakes of juniors

In an environment revealing specifically the “surprising mistakes”

➢ done even though *the developer should have known better*

- ❑ we just talked about that in class
- ❑ you got an example of how to do it right

We can learn from that, to:

- ❑ reassess our assumptions
- ❑ tune expectations
- ❑ improve teaching and mentoring
- ❑ refine code reviews

Chapters

At the end of each chapter we would have Q&A + a short discussion

Ch.1 - Intro

The settings and storyline (1)

- An advanced C++ course for undergraduates

The settings and storyline (1)

- An advanced C++ course for undergraduates
- 4th (and final) exercise in the course

The settings and storyline (1)

- An advanced C++ course for undergraduates
- 4th (and final) exercise in the course
- We worried that the exercise is a bit complicated, so we published a similar project (leveled down a bit) with a solution (“the demo project”)

The settings and storyline (1)

- An advanced C++ course for undergraduates
- 4th (and final) exercise in the course
- We worried that the exercise is a bit complicated, so we published a similar project (leveled down a bit) with a solution (“the demo project”)
- The idea was to practice them reading and using existing code and adapting it

The settings and storyline (1)

- An advanced C++ course for undergraduates
- 4th (and final) exercise in the course
- We worried that the exercise is a bit complicated, so we published a similar project (leveled down a bit) with a solution (“the demo project”)
- The idea was to practice them reading and using existing code and adapting it
- They can also learn good coding principles from the demo projects

The settings and storyline (2)

- The transition from “the demo project” to the “actual exercise” seemed to us “linear”, that is to say “technical” and “simple”

The settings and storyline (2)

- The transition from “the demo project” to the “actual exercise” seemed to us “linear”, that is to say “technical” and “simple”
- We were a bit worried that all students would just get the same outcome, with a straight A

The settings and storyline (2)

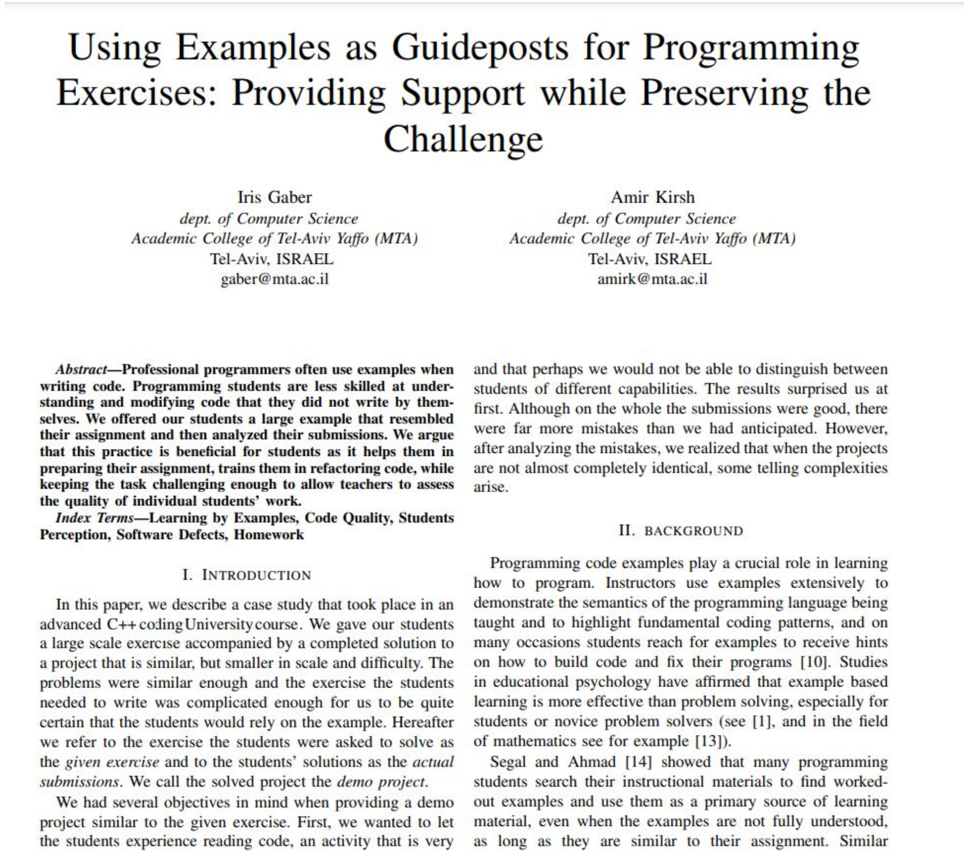
- The transition from “the demo project” to the “actual exercise” seemed to us “linear”, that is to say “technical” and “simple”
- We were a bit worried that all students would just get the same outcome, with a straight A
- Well, we were too optimistic, juniors can be very creative with all sorts of surprising mistakes

The settings and storyline (3)

- After the submission, we sent the students a questionnaire, to better understand their use of the “demo project”

The settings and storyline (3)

- After the submission, we sent the students a questionnaire, to better understand their use of the “demo project”
- Results were analyzed and published as a paper in an IEEE CS Education conference last year (winning best paper award)



Using Examples as Guideposts for Programming Exercises: Providing Support while Preserving the Challenge

Iris Gaber
dept. of Computer Science
Academic College of Tel-Aviv Yaffo (MTA)
Tel-Aviv, ISRAEL
gaber@mta.ac.il

Amir Kirsh
dept. of Computer Science
Academic College of Tel-Aviv Yaffo (MTA)
Tel-Aviv, ISRAEL
amirk@mta.ac.il

***Abstract*—Professional programmers often use examples when writing code. Programming students are less skilled at understanding and modifying code that they did not write by themselves. We offered our students a large example that resembled their assignment and then analyzed their submissions. We argue that this practice is beneficial for students as it helps them in preparing their assignment, trains them in refactoring code, while keeping the task challenging enough to allow teachers to assess the quality of individual students' work.**

***Index Terms*—Learning by Examples, Code Quality, Students Perception, Software Defects, Homework**

I. Introduction

In this paper, we describe a case study that took place in an advanced C++ coding University course. We gave our students a large scale exercise accompanied by a completed solution to a project that is similar, but smaller in scale and difficulty. The problems were similar enough and the exercise the students needed to write was complicated enough for us to be quite certain that the students would rely on the example. Hereafter we refer to the exercise the students were asked to solve as the *given exercise* and to the students' solutions as the *actual submissions*. We call the solved project the *demo project*.

We had several objectives in mind when providing a demo project similar to the given exercise. First, we wanted to let the students experience reading code, an activity that is very

and that perhaps we would not be able to distinguish between students of different capabilities. The results surprised us at first. Although on the whole the submissions were good, there were far more mistakes than we had anticipated. However, after analyzing the mistakes, we realized that when the projects are not almost completely identical, some telling complexities arise.

II. Background

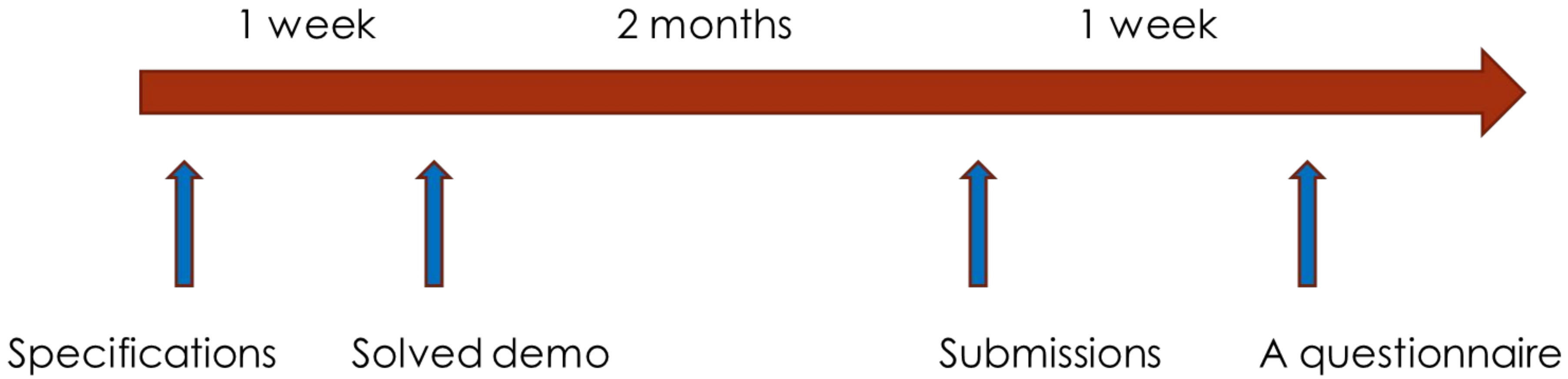
Programming code examples play a crucial role in learning how to program. Instructors use examples extensively to demonstrate the semantics of the programming language being taught and to highlight fundamental coding patterns, and on many occasions students reach for examples to receive hints on how to build code and fix their programs [10]. Studies in educational psychology have affirmed that example based learning is more effective than problem solving, especially for students or novice problem solvers (see [1], and in the field of mathematics see for example [13]).

Segal and Ahmad [14] showed that many programming students search their instructional materials to find worked-out examples and use them as a primary source of learning material, even when the examples are not fully understood, as long as they are similar to their assignment. Similar

...

https://ieeexplore.ieee.org/document/9569541

Timeline (4th exercise, towards end of semester)



End of Ch.1 - Intro

Q&A and Discussion

But please…

- Not about the idea of “using examples” (would be discussed in Ch.2)
- Not about the course itself (Ch.3 would deal with that)
- Not on the exercise or the demo project (Ch.4 and Ch.5)

Ch.2 - Coding by example

Coding by example

One of the most common coding techniques today

- Look for something similar
  - ➔ Elsewhere in my repo (my own code, others code)
  - ➔ On the web
    - ◆ In Q&A sites (e.g. Stackoverflow)
    - ◆ In an open source doing something similar

Coding by example

One of the most common coding techniques today

- Look for something similar
  - ➔ Elsewhere in my repo (my own code, others code)
  - ➔ On the web
    - ◆ In Q&A sites (e.g. Stackoverflow)
    - ◆ In an open source doing something similar

Use other code without copying (e.g. as a function / class / lib)

Take ideas and implement them in your own code

Copy snippets (1%-100%) from the other code into your own

Coding by example - then and now

Coding by example - then and now

	Then	Now
Source	A book	The web

Coding by example - then and now

	Then	Now
Source	A book	The web
Usage	Type manually into your program	Copy-Paste

Coding by example - then and now

	Then	Now
Source	A book	The web
Usage	Type manually into your program	Copy-Paste
Involvement	Higher	Lower

Coding by example - then and now

	Then	Now
Source	A book	The web
Usage	Type manually into your program	Copy-Paste
Involvement	Higher	Lower
Understanding	Usually while typing	May skip understanding

Coding by example - then and now

	Then	Now
Source	A book	The web
Usage	Type manually into your program	Copy-Paste
Involvement	Higher	Lower
Understanding	Usually while typing	May skip understanding
Potential bugs	Typos	Copy-Paste errors

Common mistakes while coding by example

Common mistakes while coding by example (1)

Not adapting the example to our exact needs

- leaving unnecessary parts of the example
- not aligning to the target’s coding standards
- hurting conceptual integrity of our design
- ignoring better mashup possibilities

Common mistakes while coding by example (2)

Not choosing the right example

- the xy problem
- using an old code example
- ignoring better options (taking the first example that seems to fit)
- copying code from the question (e.g. in Stackoverflow) instead of from the answers

Common mistakes while coding by example (3)

Adapting the example wrongly or without a need

- having wrong intuition that a certain part of the code is not needed
- trying to be too cute
- trying to adapt the code to the existing coding standards or design, creating bugs along the process

Fast and Slow Thinking

Kahneman's work, see: https://suebehaviouraldesign.com/kahneman-fast-slow-thinking/

System 1

- FAST
- Unconscious, automatic, effortless
- WITHOUT self-awareness or control "What you see is all there is."
- ROLE: Assesses the situation, delivers updates
- Does 98% of all our thinking

System 2

- SLOW
- Deliberate and conscious, effortful, controlled mental process, rational thinking
- WITH self-awareness or control, logical and skeptical
- ROLE: seeks new/missing information, makes decisions
- Does 2% of all our thinking

* thanks for Mathias Schulz for discussing this with me

Fast and Slow Thinking

Which one do we use when assimilating code examples?
And when writing our own code?

System 1

- FAST
- Unconscious, automatic, effortless
- WITHOUT self-awareness or control “What you see is all there is.”
- ROLE: Assesses the situation, delivers updates
- Does 98% of all our thinking

System 2

- SLOW
- Deliberate and conscious, effortful, controlled mental process, rational thinking
- WITH self-awareness or control, logical and skeptical
- ROLE: seeks new/missing information, makes decisions
- Does 2% of all our thinking

* thanks for Mathias Schulz for discussing this with me

Is an external code snippet a “black-box”?

We are used to modularity.

As systems become bigger we feel more comfortable treating part of the system as a black-box.

But a code snippet is NOT a black-box! It becomes part of your code.

How do you know that this regex works well?

- *Well, I took it from Stackoverflow*
- *I treat it as a black-box*

Should an external code snippet be tested?

Sure!

Even before you integrate it into your own code!

Do you do it?

When it doesn’t work, what would you do?

Juniors tend to use “trial and error”...
(Let’s move this loop here and that ‘if’ there, maybe it would help).

Instead of:

- adding debug logs / printouts, breakpoints - analysis!
- trying to have a theory of what happens and check it (debugging in a methodological way)
- divide and conquer - go back and forth carefully to the latest position where it works

But it does work!

Juniors tend to feel secure when code works.

Experienced C++ developers know that “working” is not a guarantee.

```cpp
int main() {
    std::vector<int> vec = {1, 2};
    vec.push_back(3);
    vec[3] = 42; // index out of bounds, but it works!
    std::cout << vec[3] << '\n';
    std::cout << vec.capacity() << '\n';
}
```

https://ub.godbolt.org/z/EbrTb3qMb

But it does work!

Juniors tend to feel secure when code works.

Experienced C++ developers know that “working” is not a guarantee.

```
int main() {
    std::vector<int> vec = {1, 2};
    vec.push_back(3);
    vec[3] = 42; // index out of bounds, but it works!
    std::cout << vec[3] << '\n';
    std::cout << vec.capacity() << '\n';
}
```

this is true for any SW language, not only for C++, but it’s much more relevant in C++ where “it’s UB” is the answer for 7.1% of questions

https://ub.godbolt.org/z/EbrTb3qMb

But it does work!

Juniors tend to feel secure when code works.

Experienced C++ developers know that “working” is not a guarantee.

```
int main() {
    std::vector<int> vec = {1, 2};
    vec.push_back(3);
    vec[3] = 42; // index out of bounds, but it works!
    std::cout << vec[3] << '\n';
    std::cout << vec.capacity() << '\n';
}
```

this is true for any SW language, not only for C++, but it’s much more relevant in C++ where “it’s UB” is the answer for 7.1% of questions

↑

don’t take this number too seriously, I just made it up

https://ub.godbolt.org/z/EbrTb3qMb

Examples are a bit addicting

Like intellisense and auto completion are.

We are slowly losing our ability to implement code totally on our own.

(And then comes Copilot, Tabnine etc.)

But do not go with NIH approach

No need to re-implement existing things that work well.
(You have a working example, use it!)

Even if you think you would do better, usually you would not.

I just feel I want to write this piece on my on - is not an excuse.

A legal note

Watch the license when you copy massive snippets of code.

Copying GPL code snippets might require you to be GPL as well.
It is problematic to copy LGPL code snippet.
…

(This is not the purpose of this talk, but should be raised).

Coding by example

It’s more complicated than one may think

A copied code snippet *cannot* be treated as a black-box

End of Ch.2 - Coding by example

Q&A and Discussion

Specifically:

- Is using examples a good or a bad thing? WDYT?
- Any other common mistakes of using code examples you would like to add?
- Ways to avoid common mistakes while coding by example?

Ch.3 - A few words about the C++ course

The course

- 3rd year CS undergraduates, non-compulsory course, 63 students
- After Java (as their OOP lang) and C (as their “System programming lang”)
- Course includes:
    - ➔ the basic C++ syntax
    - ➔ rvalue and move semantics
    - ➔ templates
    - ➔ lambda expressions
    - ➔ std containers and algorithms
    - ➔ smart pointers
    - ➔ concurrency and multithreading
    - ➔ massive 4 code exercises

The course

16% women

- 3rd year CS undergraduates, non-compulsory course, 63 students
- After Java (as their OOP lang) and C (as their "System programming lang")
- Course includes:
  - ➔ the basic C++ syntax
  - ➔ rvalue and move semantics
  - ➔ templates
  - ➔ lambda expressions
  - ➔ std containers and algorithms
  - ➔ smart pointers
  - ➔ concurrency and multithreading
  - ➔ massive 4 code exercises

End of Ch.3 - The course

Any questions?

Ch.4 - The exercise

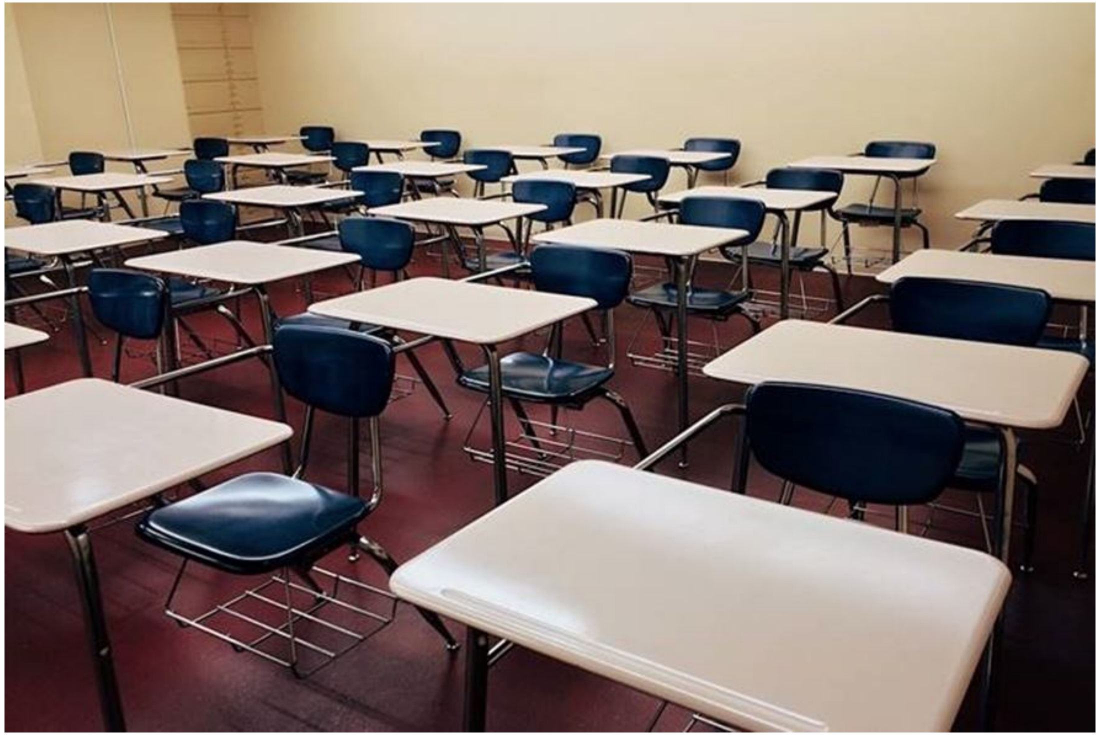
Implement an API for a Ship class with dimensions X * Y * Height, holding containers, including: iterators, views and find algorithms

Link to exercise – Note: you can use std containers and algorithms

End of Ch.4 - The exercise

Any questions?

Ch.5 - The demo project



Implement an API for an ExamsHall class with dimensions X * Y, holding examinees, including: iterators, views and find algorithms

Link to requirements – Link to provided demo solution

End of Ch.5 - The demo project

Questions? Comments?

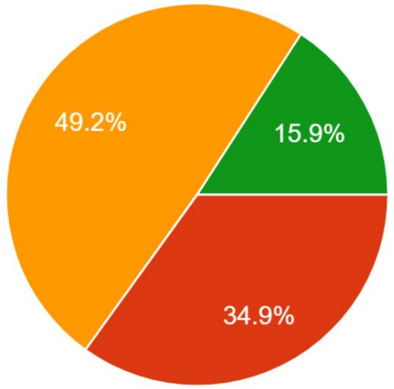
Ch.6 - Defects and failures

Just before…

Did they actually use the demo project?

I used the Exams Hall example as a basis for my solution

63 responses



● No

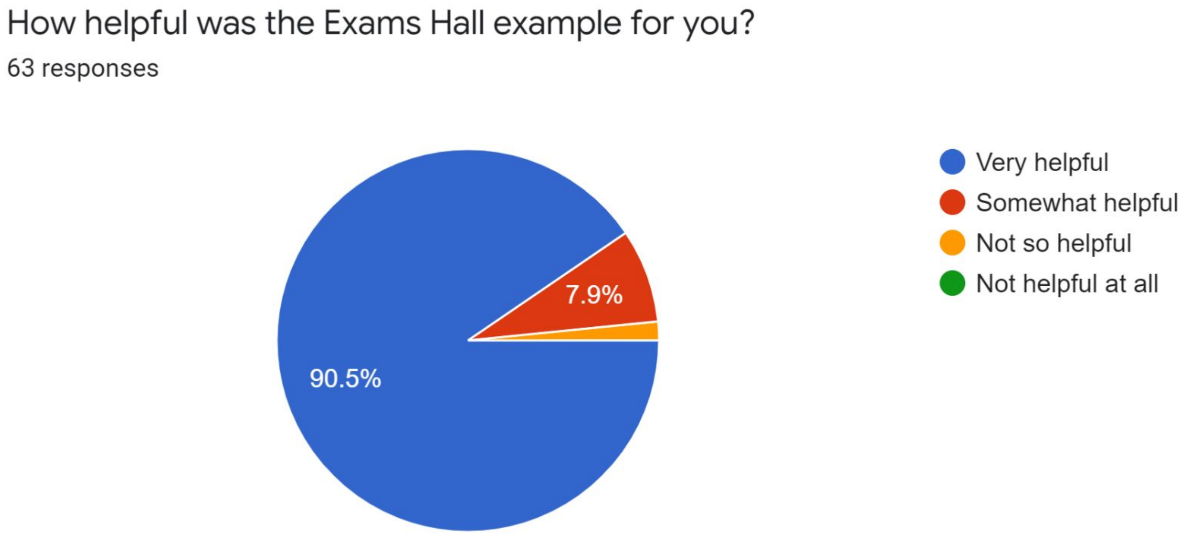
● Yes - I worked directly on the Exams Hall code example and made changes on it to fit the Ship's exercise

● Yes - I copied relevant snippets from the Exams Hall code example to my Ship's exercise

● Partially - I looked at the Exams Hall example and used some parts, but wrote most of the Ship's exercise on my own

Just before…

Was the demo project helpful?



Categorizing the mistakes

Categorizing the mistakes

➔ Not using code or principles from the demo project

Categorizing the mistakes

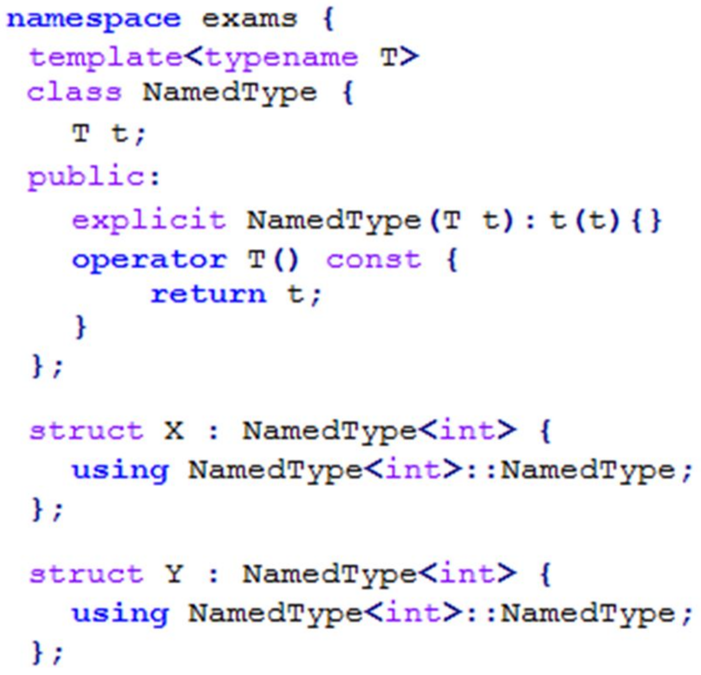
- ➔ Not using code or principles from the demo project
- ➔ Ignoring demo project inadequacies for real project

Categorizing the mistakes

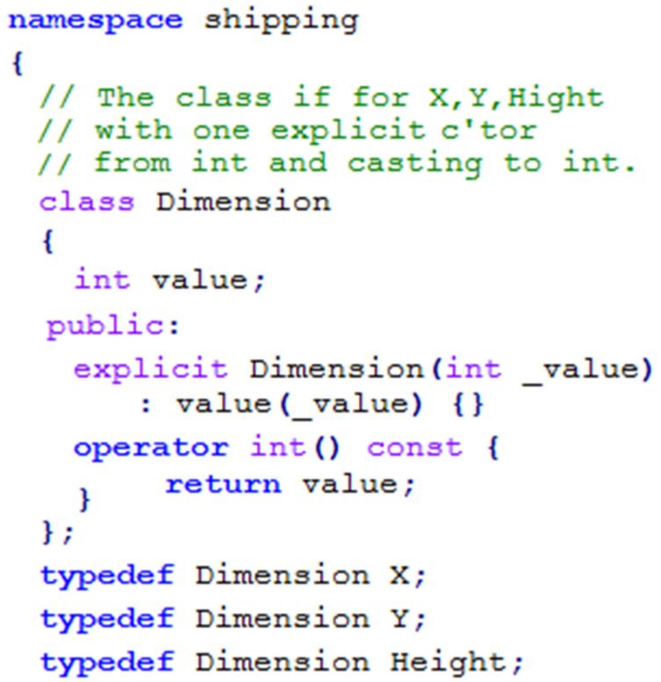
- ➔ Not using code or principles from the demo project
- ➔ Ignoring demo project inadequacies for real project
- ➔ Other mistakes, not related to the demo project

Not using code or principles from the demo project

Not using code or principles from the demo project

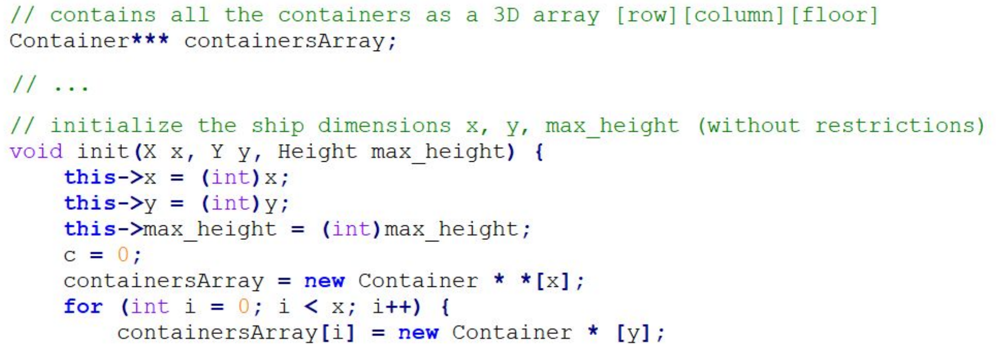


```cpp
namespace exams {
 template<typename T>
 class NamedType {
   T t;
 public:
   explicit NamedType(T t):t(t){}
   operator T() const {
       return t;
   }
 };

 struct X : NamedType<int> {
   using NamedType<int>::NamedType;
 };

 struct Y : NamedType<int> {
   using NamedType<int>::NamedType;
 };
```

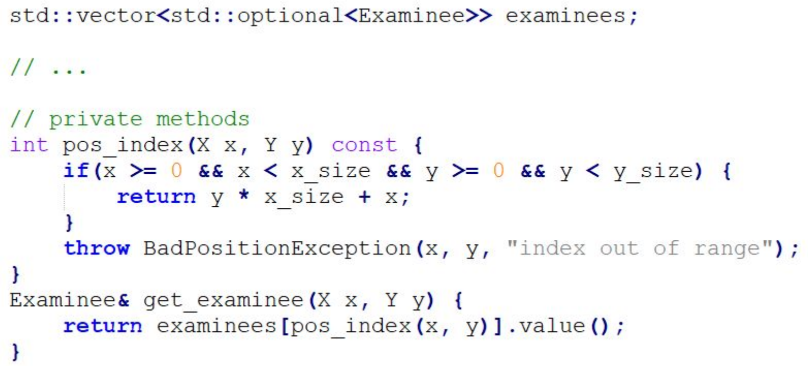


```cpp
namespace shipping
{
  // The class if for X,Y,Hight
  // with one explicit c'tor
  // from int and casting to int.
  class Dimension
  {
    int value;
   public:
    explicit Dimension(int _value)
        : value(_value) {}
    operator int() const {
          return value;
    }
  };
  typedef Dimension X;
  typedef Dimension Y;
  typedef Dimension Height;
```

Not using code or principles from the demo project



```
// contains all the containers as a 3D array [row][column][floor]
Container*** containersArray;

// ...

// initialize the ship dimensions x, y, max_height (without restrictions)
void init(X x, Y y, Height max_height) {
    this->x = (int)x;
    this->y = (int)y;
    this->max_height = (int)max_height;
    c = 0;
    containersArray = new Container * *[x];
    for (int i = 0; i < x; i++) {
        containersArray[i] = new Container * [y];
```

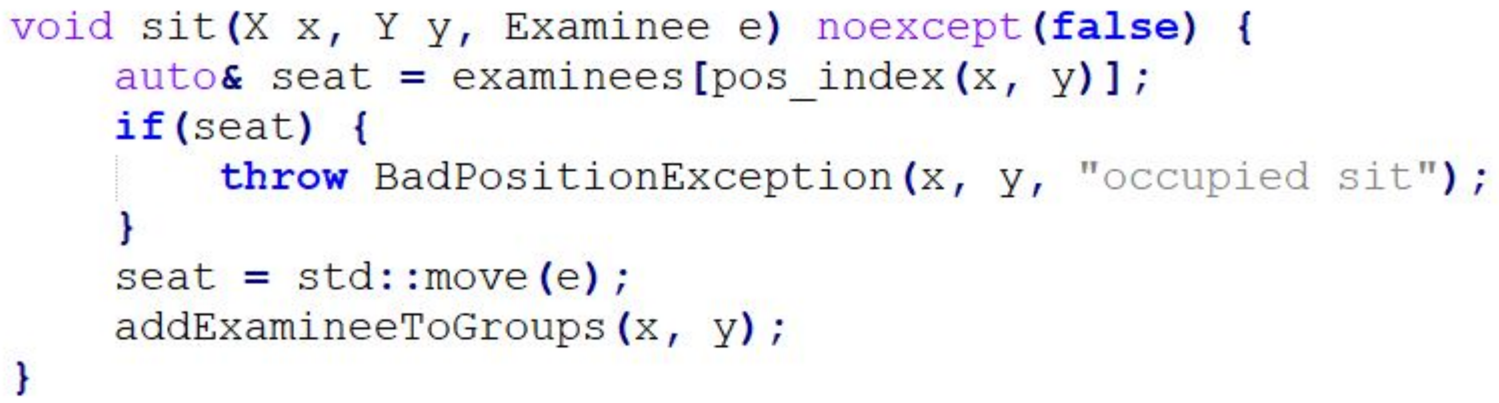


```
std::vector<std::optional<Examinee>> examinees;

// ...

// private methods
int pos_index(X x, Y y) const {
    if(x >= 0 && x < x_size && y >= 0 && y < y_size) {
        return y * x_size + x;
    }
    throw BadPositionException(x, y, "index out of range");
}
Examinee& get_examinee(X x, Y y) {
    return examinees[pos_index(x, y)].value();
}
```

Not using code or principles from the demo project

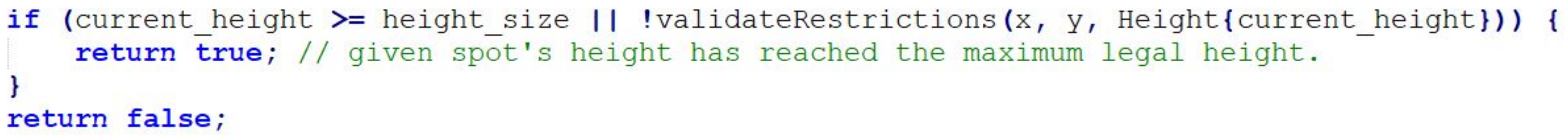
```
/**
 * @throws BadShipOperationException
 */
void load(X x, Y y, Container c) noexcept(false) {
    if (!(0 <= x && x <= maxX) || !(0 <= y && y <= maxY)) {
        throw BadShipOperationException("Invalid position x=" + std::to_string(x) + ", y=" + std::to_string(y));
    }

    if (storage[x][y].size() == maxHeightByPos[x][y]) {
        throw BadShipOperationException("Ship is full at position x=" + std::to_string(x) + ", y=" + std::to_string(y));
    }

    storage[x][y].push_back(c);
}

/**
 * @throws BadShipOperationException
 */
Container unload(X x, Y y) noexcept(false) {
    if (!(0 <= x && x <= maxX) || !(0 <= y && y <= maxY)) {
        throw BadShipOperationException("Invalid position x=" + std::to_string(x) + ", y=" + std::to_string(y));
    }

    if (storage[x][y].size() == 0) {
        throw BadShipOperationException("Ship is empty at position x=" + std::to_string(x) + ", y=" + std::to_string(y));
    }

    Container container = std::move(storage[x][y].back());
    storage[x][y].pop_back();
    return container;
}
```

Not using code or principles from the demo project



```cpp
void sit(X x, Y y, Examinee e) noexcept(false) {
    auto& seat = examinees[pos_index(x, y)];
    if(seat) {
        throw BadPositionException(x, y, "occupied sit");
    }
    seat = std::move(e);
    addExamineeToGroups(x, y);
}
```

Not using code or principles from the demo project



```
if (current_height >= height_size || !validateRestrictions(x, y, Height{current_height})) {
    return true; // given spot's height has reached the maximum legal height.
}
return false;
```

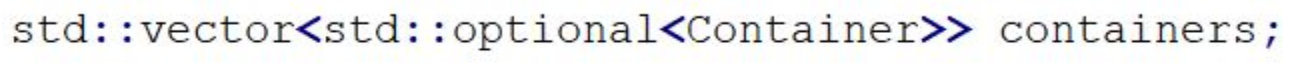
Not using code or principles from the demo project

Other:

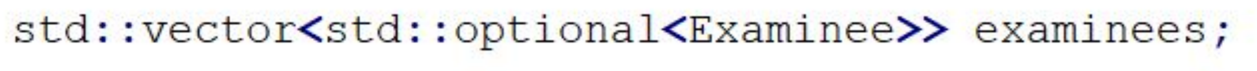
- ➔ Duplicating code that was presented in the demo project in a proper helper
- ➔ Initializing in constructor body instead of in constructor init-list
- ➔ Looping on a vector to initialize it to zeros
- ➔ Holding a unique_ptr to a vector (3D), with no need - similar vector (2D) was held as a value in the demo project
- ➔ Inventing new naming conventions
- ➔ Exposing data members in the public
- ➔ Using old style “throw()” instead of “noexcept”

Ignoring demo project inadequacies for real project

Ignoring demo project inadequacies for real project

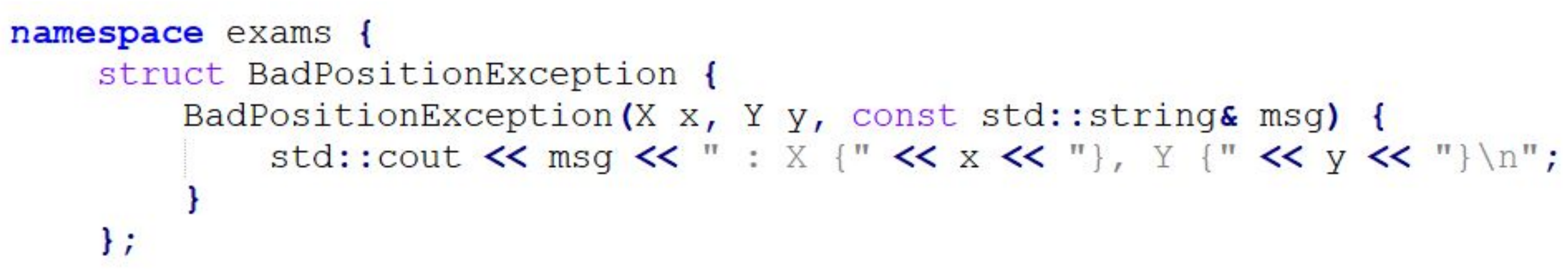


```cpp
std::vector<std::optional<Container>> containers;
```



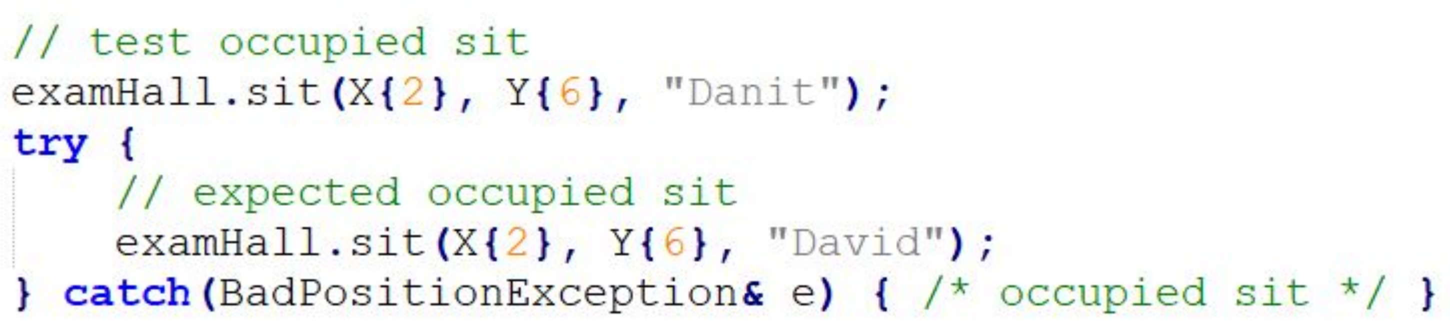
```cpp
std::vector<std::optional<Examinee>> examinees;
```

Ignoring demo project inadequacies for real project



```
namespace exams {
    struct BadPositionException {
        BadPositionException(X x, Y y, const std::string& msg) {
            std::cout << msg << " : X {" << x << "}, Y {" << y << "}\n";
        }
    };
```

Ignoring demo project inadequacies for real project



```
// test occupied sit
examHall.sit(X{2}, Y{6}, "Danit");
try {
    // expected occupied sit
    examHall.sit(X{2}, Y{6}, "David");
} catch(BadPositionException& e) { /* occupied sit */ }
```

Other mistakes, not related to the demo project

- ➔ Some of those who allocated memory forgot a destructor or forgot to implement or delete the copy constructor and assignment operator
- ➔ const correctness
- ➔ std::move, stealing from objects that are still in use
- ➔ Unnecessary copies of objects, including entire collections, passing by-value without a reason (and then, not moving from it)
- ➔ Code duplication
- ➔ All sorts of algorithmic inefficiencies
- ➔ Implementation doesn’t fit the requirements

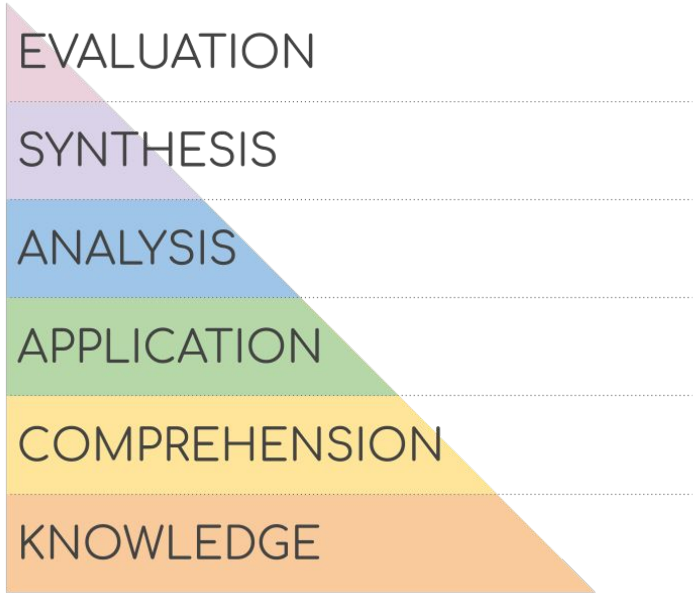
End of Ch.6 - Defects and failures

Questions? Comments?

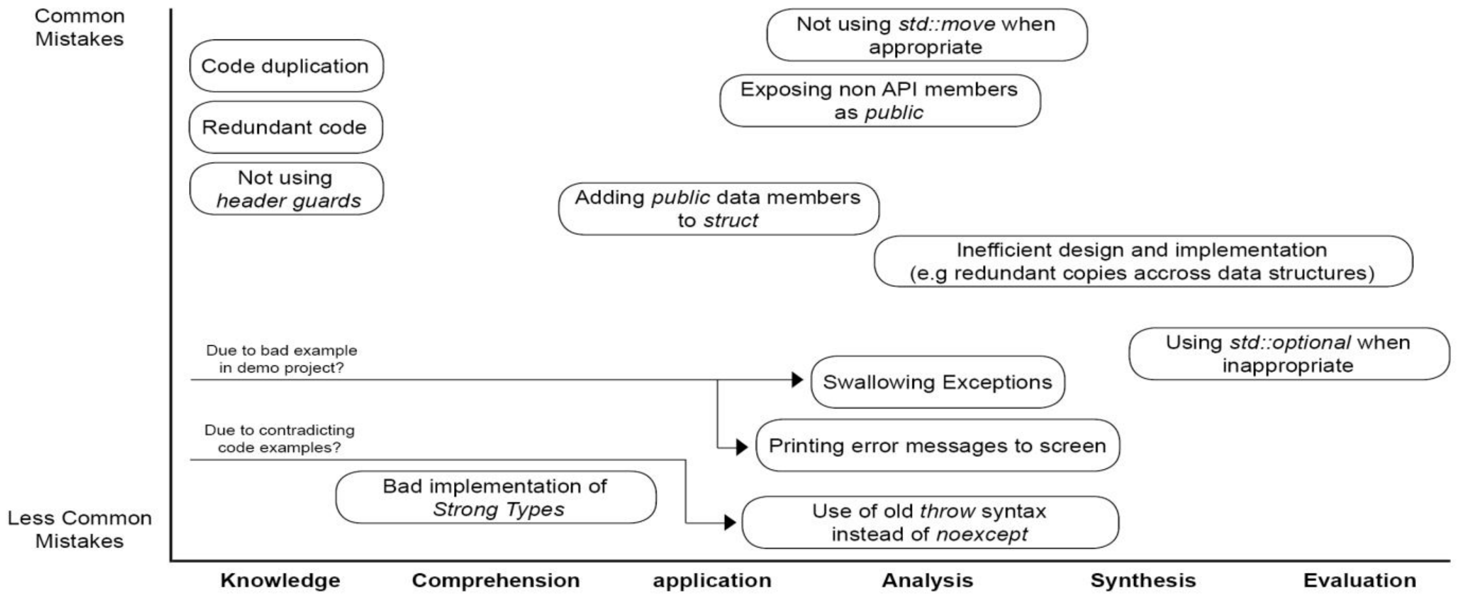
Ch.7 - Conclusions and recommendations

Mistakes are in all levels of Blum's taxonomy

Blum's cognitive taxonomy levels



Mistakes are in all levels of Blum's taxonomy


Common Mistakes
Less Common Mistakes
Code duplication
Redundant code
Not using *header guards*
Not using *std::move* when appropriate
Exposing non API members as *public*
Adding *public* data members to *struct*
Inefficient design and implementation (e.g redundant copies accross data structures)
Using *std::optional* when inappropriate
Due to bad example in demo project?
Swallowing Exceptions
Printing error messages to screen
Due to contradicting code examples?
Bad implementation of *Strong Types*
Use of old *throw* syntax instead of *noexcept*
Knowledge
Comprehension
application
Analysis
Synthesis
Evaluation

Students' mistakes across Bloom's taxonomy

Using code examples is harder than appears

Requires practicing

+ Special attention in code review

- ➔ What does the code suppose to do?
- ➔ How was it tested?
- ➔ Does it fit our design and coding conventions?
- ➔ Is the license approved by the company? (if more than few code lines were copied)
- ➔ If you were asked to put your name on this piece of **** would you?

code, of course
what were you
thinking of?

On putting @author tags in code

On putting @author tags in code

I’m in favor of adding @author notes on file, methods and changed code

This contradicts google style guide advice
https://google.github.io/styleguide/cppguide.html#File_Comments

But I recommend that, for 2 reasons:

[1]

In many cases, the contributor listed in source control is not the actual contributor (private merge and integrations from other branch)

[2]

Sense of responsibility, accountability and professional pride

Juniors’ approach to problems to beware of

Juniors’ approach to problems to beware of

Let’s not ask, it may show we don’t understand

Juniors’ approach to problems to beware of

It’s good I didn’t ask!

I got it done on my own!

Juniors’ approach to problems to beware of

It’s good I didn’t ask!

I got it done on my own!

Source: https://www.ba-bamail.com/content.aspx?emailid=36683

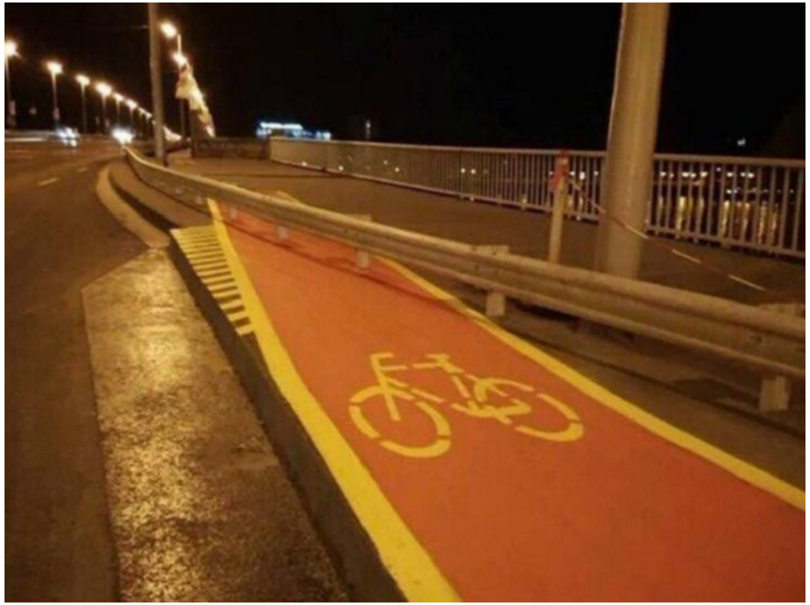
Juniors’ approach to problems to beware of

It fits the requirements.
I think.

Juniors' approach to problems to beware of

It fits the requirements.

I think.



Source: https://www.ba-bamail.com/content.aspx?emailid=36683

Juniors’ approach to problems to beware of

What’s so bad with some undefined behavior?

https://memegenerator.net/instance/63896485/spongebob-rainbow-undefined-behavior

Juniors' approach to problems to beware of

I think I got it right now.

Juniors’ approach to problems to beware of

I think I got it right now.

Source: https://www.ba-bamail.com/content.aspx?emailid=36683

Juniors’ approach to problems to beware of

I think I fixed it.

Juniors’ approach to problems to beware of

I think I fixed it.

Source: https://www.ba-bamail.com/content.aspx?emailid=36683

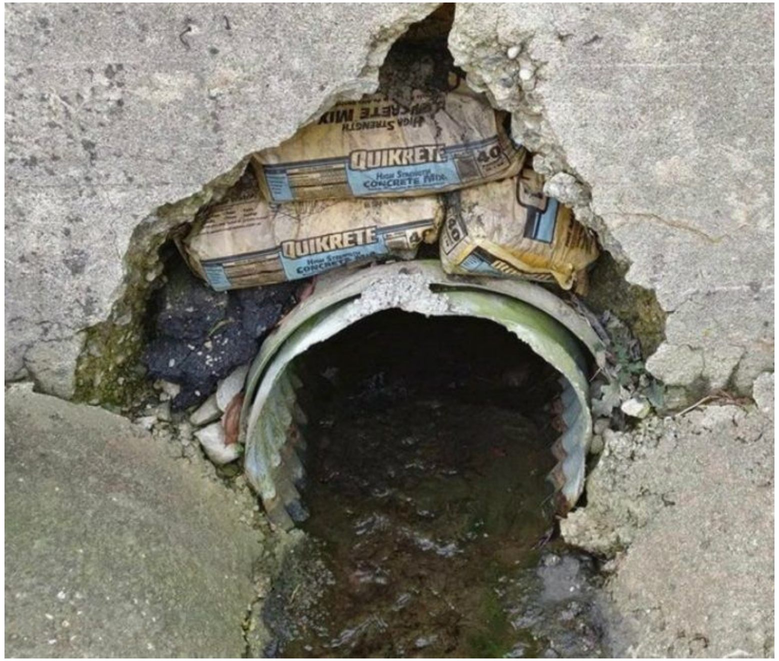
We are not far away from the end

Juniors’ approach to problems to beware of

I think I fixed it.

Juniors’ approach to problems to beware of

I think I fixed it.



Source: https://www.ba-bamail.com/content.aspx?emailid=36683

Juniors’ approach to problems to beware of

I think I got it right now.

Juniors’ approach to problems to beware of

I think I got it right now.

Source: https://www.ba-bamail.com/content.aspx?emailid=36683

Juniors’ approach to problems to beware of

Oops. Something is broken.
Let’s document it, so someone would take care of it.

Juniors’ approach to problems to beware of

Oops. Something is broken. Let’s document it, so someone would take care of it.

Emergency Phone is Out of Service
Please do not have an emergency at this location

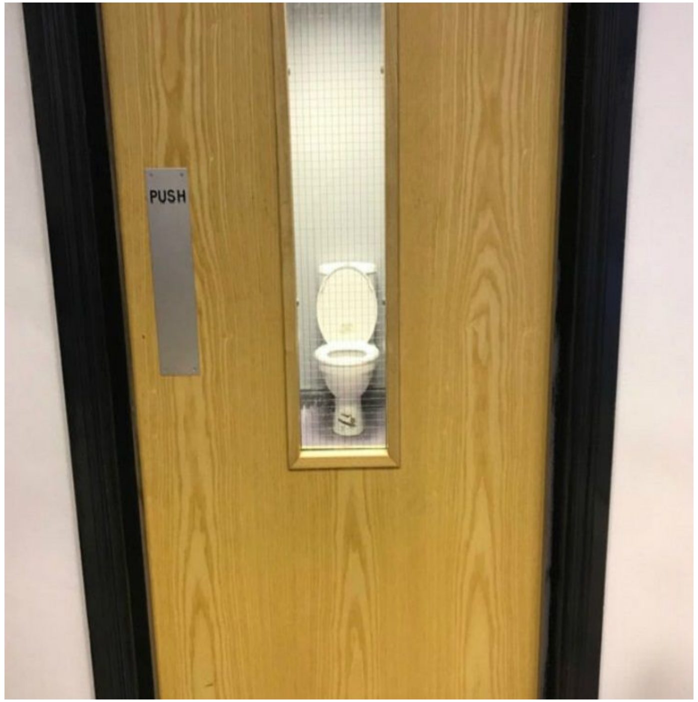
Source: https://www.ba-bamail.com/content.aspx?emailid=36683

Juniors’ approach to problems to beware of

Let’s put things in the public.
Transparency makes things more accessible.

Juniors’ approach to problems to beware of

Let’s put things in the public.
Transparency makes things more accessible.



Source: https://www.ba-bamail.com/content.aspx?emailid=36683

Any questions before we conclude?

Final thing before we depart…

CPP Game Show
from Incredibuild
25:00
C++
CPP GAME SHOW
Amir
Daisy
Dima
Vittorio
Kilian