Admin

- Take-home Midterm due.
- For next week, please listen to soft skills episode 446
  - (we did this one already)

Chapter 2 of pragmatic programmer

- If we haven't talked about it yet, do so here.

Sprint Retrospective

- If not done earlier, lets do the sprint retrospective.

Development Methodologies

- What are some common development methodologies?

Development Methodologies

- What are some common development methodologies?
  - Two most well known today are likely Agile and Waterfall
  - Historical xtreme programming etc

Development Methodologies

- What are some common development methodologies?
  - Two most well known today are likely Agile and Waterfall
  - Lets describe Waterfall
  - And Agile
  - And tell why you would use each in the 2020s

Self Documenting code?

- Decades long debate:
- Can code be truly self documenting?
  - Do we really need comments?
  - After all this can be the ultimate DRY violation

Self Documenting code?

- Decades long debate:
- Can code be truly self documenting?
  - Do we really need comments?
  - After all this can be the ultimate DRY violation
- My phd advisor's take.
-

Self Documenting code?

- Decades long debate:
- Can code be truly self documenting?
  - Do we really need comments?
  - After all this can be the ultimate DRY violation
- My phd advisor's take.
- The ‘big 3’ books’ takes
  - Make code as self documenting as possible – any comments needed is a failure of clear code/ is deodorant for ‘code smells’
- My take:

Self Documenting code?

- Decades long debate:
- Can code be truly self documenting?
  - Do we really need comments?
  - After all this can be the ultimate DRY violation
- My phd advisor's take.
- The ‘big 3’ books’ takes
  - Make code as self documenting as possible – any comments needed is a failure of clear code/ is deodorant for ‘code smells’
- My take:
  - As per books above, except that dynamically typed languages need a little more comment TLC.

Programming Style

- Programming style is important for self-documenting code
  - Different languages have different style guides
  - More on that later in the semester
  - But follow the style guide
  - Or better yet use an auto formatter
    - Gofmt
    - Rustfmt
    - Python: just like Henry Ford: “any color you like ...”

Black

Comments

- In CS1 and CS2 (comp151 and comp152 here) instructors often tell you to comment your code
  - Why?

Comments

- In CS1 and CS2 (comp151 and comp152 here) instructors often tell you to comment your code
  - Why?
  - Often so we can tell what you thought you were writing.
  - If you look back at the code from that era of your lives, you'll often find that you did not write what you think you did

Comments in production code

- In production code, comments should be relatively rare, and very meaningful.
- “Comments often are used as a deodorant.”
  - — Martin Fowler and Kent Beck, Refactoring, page 87
  - This book is from forever ago (1999ish)
- Comments are often deodorant for code smells, so mostly make your code better

The problem with comments in real life

- Orphaned comments
  - Page 54 of clean code
- Comment rot (page 785 in code complete)
  - Comment written – then code is changed – but not comment
  - To misquote Mark Twain:

The problem with comments in real life

- Orphaned comments
  - Page 54 of clean code
- Comment rot (page 785 in code complete)
  - Comment written – then code is changed – but not comment
- To misquote Mark Twain: there are three kinds of lies
  - Lies! Damn Lies and old Comments!!

So what comments do you still need?

- Legal comments
  - These are barely for the code anyway
- Explaining why you chose to do it some way
- ToDo
  - These are comments meant to go away
  - explain why something doesn't make sense

So what comments do you still need?

- Authors argue for
  - Acceptable range of values for a variable
    - Though this should also be checked by your automated tests today.
  - Limitations on input data
    - Same as above – also enforce with tests
  - Bits in a bit mask
    - This falls under the obscure stuff that needs to be tracked
    - What do I mean by Bit mask?

So what comments do you still need?

- Warnings:
  - Warn of the consequences of code
  - Don't do this unless …
- Informative for bizarre needs
  - Fixing a bug in a library that you don't own example
- Documentation comments
  - Javadoc comments
  - Python API documentation strings/comments

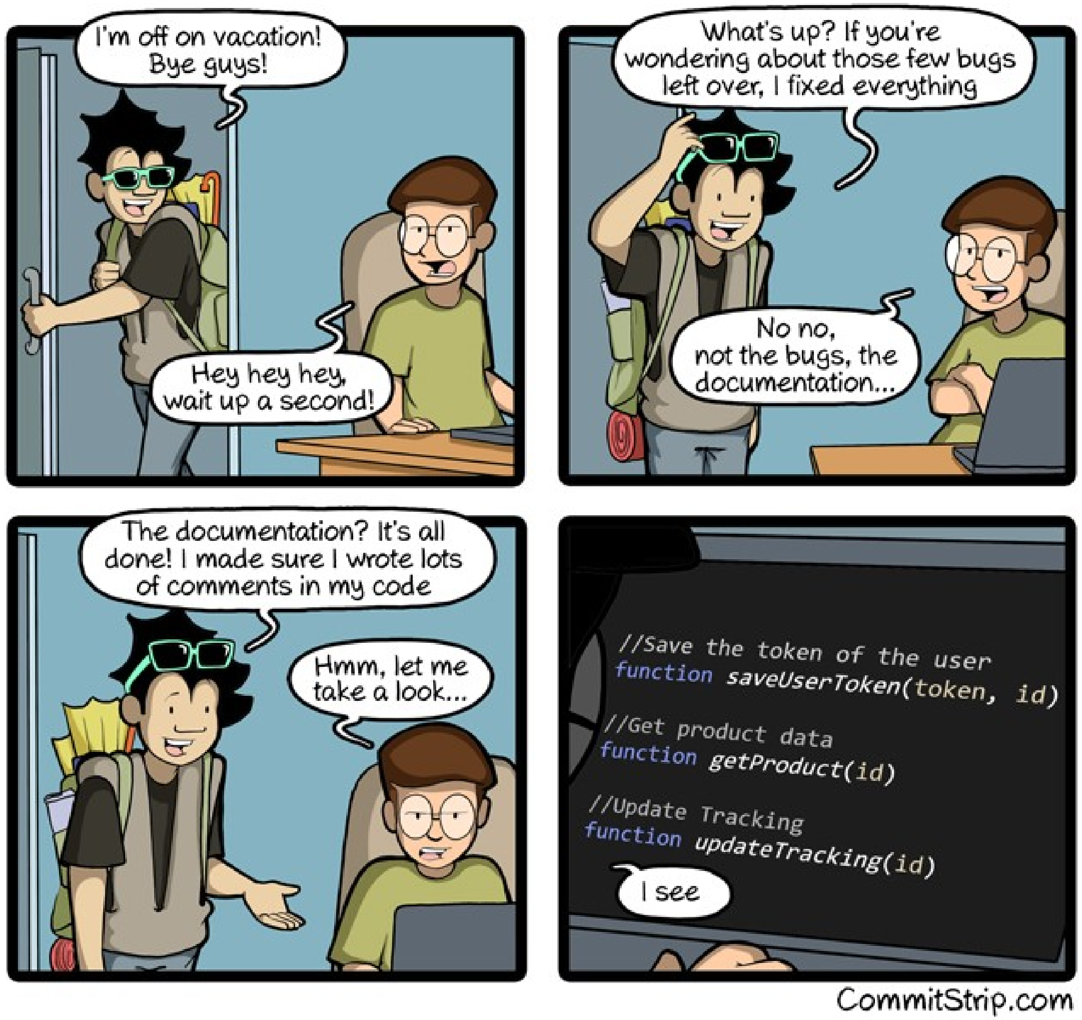
My favorite comment quote

- I've seen this in more than one place,
  - "code should explain how (and what), the comments should explain why"
  - Too true.

Comments to avoid I

- Some comments are bad
  - If someone else reading the comment has to read code elsewhere to know what the comment means
  - Book: mumbling
- Redundant comments
  - j=j+1; //increment j (huge DRY violation)
  - Uh-huh because I failed comp151
  - Note book has larger definition
  - But awesome example on page 64 (Clean Code)(and next slide)

Dry Violation comments


I'm off on vacation! Bye guys!
Hey hey hey, wait up a second!
What's up? If you're wondering about those few bugs left over, I fixed everything
No no, not the bugs, the documentation...
The documentation? It's all done! I made sure I wrote lots of comments in my code
Hmm, let me take a look...
I see
CommitStrip.com

```
//Save the token of the user
function saveUserToken(token, id)

//Get product data
function getProduct(id)

//Update Tracking
function updateTracking(id)
```

Less than ideal comments

- Some really old projects (and there are still lots in production) have historical comments that are/were required
  - Mandated comments
    - Some employers (DoD was famous for this) require certain structures and strictures to be followed
    - Maybe useless, but must have them
  - Journal comments
    - Make a comment at top of file everytime a change is made to the code
    - This is what commit comments in the version control system are for today

Comments to avoid II

- The pasto
  - Copy – pasted code with comments
  - Change code (clean code 66)
  - Hello comment rot.
- Commented out code
  - Now we do this in academia
  - But for code that will go to production:
    - Commented out code stays forever
    - Too important to delete?
    - Worse – multiline commented code

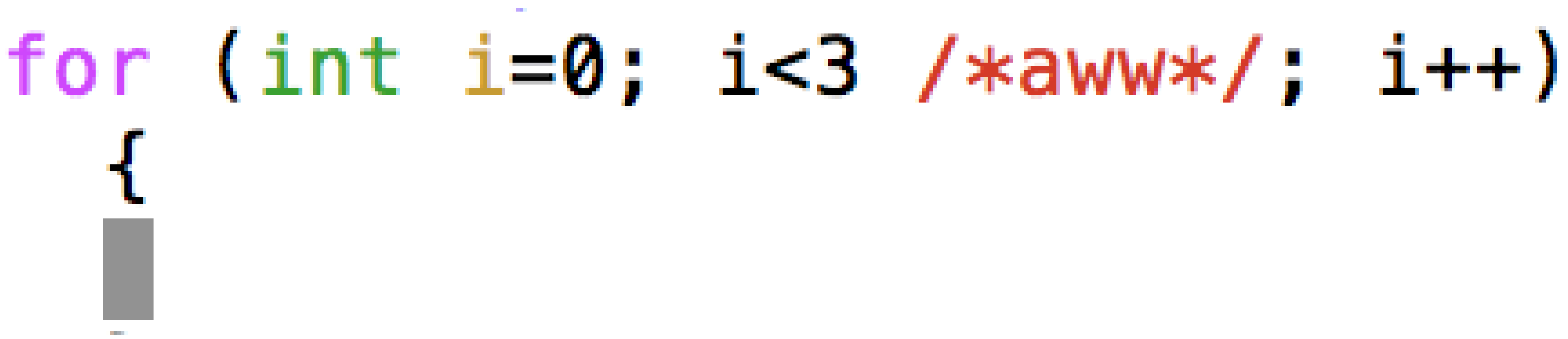
Comments to avoid III

- Cute comments:
  - Eg from stack overflow:
  - /**
  - * For the brave souls who get this far: You are the chosen ones,
  - * the valiant knights of programming who toil away, without rest,
  - * fixing our most awful code. To you, true saviors, kings of men,
  - * I say this: never gonna give you up, never gonna let you down,
  - * never gonna run around and desert you. Never gonna make you cry,
  - * never gonna say goodbye. Never gonna tell a lie and hurt you.
  - */

Comments to avoid IV

- And bad for several reasons: (what are they?)
  - //
  - // Dear maintainer:
  - //
  - // Once you are done trying to 'optimize' this routine,
  - // and have realized what a terrible mistake that was,
  - // please increment the following counter as a warning
  - // to the next guy:
  - //
  - // total_hours_wasted_here = 42
  - //

Again, don't be cute



```
for (int i=0; i<3 /*aww*/; i++)
    {
```

Patting yourself on the back comments

- From Code Complete book (pg 792)
  - MOV AX, 723h ; R. I. P. L. V. B.
    - RIP Ludvig Van Beethoven
    - Died 1827(dec) which is?

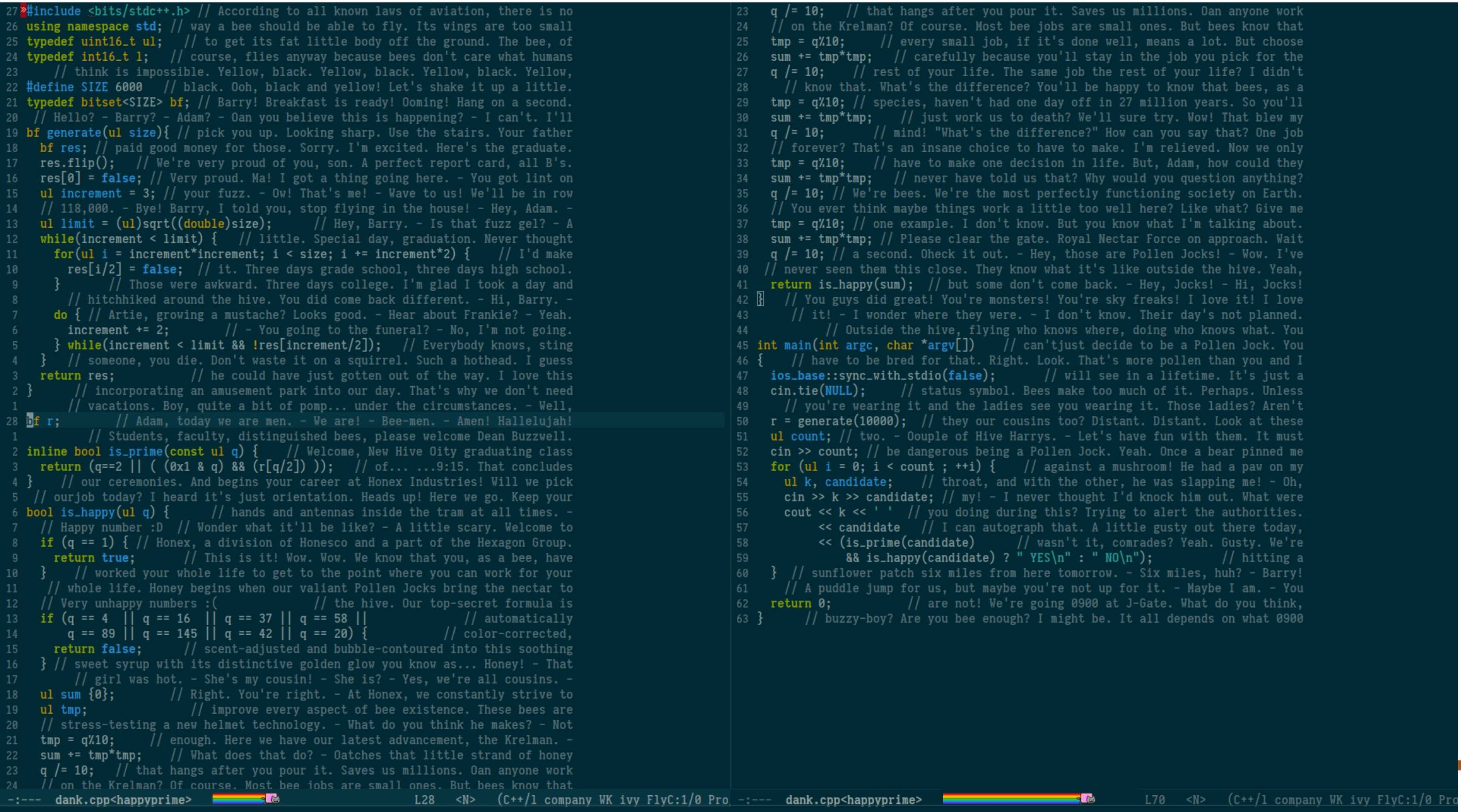
Patting yourself on the back comments

- From Code Complete book (pg 792)
  - MOV AX, 723h ; R. I. P. L. V. B.
    - RIP Ludvig Van Beethoven
    - Died 1827(dec) which is?
    - Yup 723 hex
    - What does Beethoven have to do with anything? Nothing!

Comment Relevence

- Make comments relevant
  - stop(); // Hammertime!
  - Probably was very cute 15(World of Warcraft)-30 (original song release) years ago
  - “You see kids...” (one day this meme will be as obsolete as your parents music)

Got this one in 2018 (Bee movie 10th anniversary)



```cpp
#include <bits/stdc++.h> // According to all known laws of aviation, there is no
using namespace std; // way a bee should be able to fly. Its wings are too small
typedef uint16_t ul;   // to get its fat little body off the ground. The bee, of
typedef int16_t l;   // course, flies anyway because bees don't care what humans
     // think is impossible. Yellow, black. Yellow, black. Yellow, black. Yellow,
#define SIZE 6000   // black. Ooh, black and yellow! Let's shake it up a little.
typedef bitset<SIZE> bf; // Barry! Breakfast is ready! Ooming! Hang on a second.
 // Hello? - Barry? - Adam? - Oan you believe this is happening? - I can't. I'll
bf generate(ul size){ // pick you up. Looking sharp. Use the stairs. Your father
  bf res; // paid good money for those. Sorry. I'm excited. Here's the graduate.
  res.flip();   // We're very proud of you, son. A perfect report card, all B's.
  res[0] = false; // Very proud. Ma! I got a thing going here. - You got lint on
  ul increment = 3; // your fuzz. - Ow! That's me! - Wave to us! We'll be in row
  // 118,000. - Bye! Barry, I told you, stop flying in the house! - Hey, Adam. -
  ul limit = (ul)sqrt((double)size);      // Hey, Barry. - Is that fuzz gel? - A
  while(increment < limit) {   // little. Special day, graduation. Never thought
    for(ul i = increment*increment; i < size; i += increment*2) {    // I'd make
      res[i/2] = false;  // it. Three days grade school, three days high school.
    }          // Those were awkward. Three days college. I'm glad I took a day and
      // hitchhiked around the hive. You did come back different. - Hi, Barry. -
    do { // Artie, growing a mustache? Looks good. - Hear about Frankie? - Yeah.
      increment += 2;         // - You going to the funeral? - No, I'm not going.
    } while(increment < limit && !res[increment/2]);   // Everybody knows, sting
  }   // someone, you die. Don't waste it on a squirrel. Such a hothead. I guess
  return res;           // he could have just gotten out of the way. I love this
}       // incorporating an amusement park into our day. That's why we don't need
      // vacations. Boy, quite a bit of pomp... under the circumstances. - Well,
bf r;        // Adam, today we are men. - We are! - Bee-men. - Amen! Hallelujah!
        // Students, faculty, distinguished bees, please welcome Dean Buzzwell.
inline bool is_prime(const ul q) {    // Welcome, New Hive Oity graduating class
  return (q==2 || ( (0x1 & q) && (r[q/2]) ));   // of... ...9:15. That concludes
}    // our ceremonies. And begins your career at Honex Industries! Will we pick
 // ourjob today? I heard it's just orientation. Heads up! Here we go. Keep your
bool is_happy(ul q) {      // hands and antennas inside the tram at all times. -
  // Happy number :D  // Wonder what it'll be like? - A little scary. Welcome to
  if (q == 1) { // Honex, a division of Honesco and a part of the Hexagon Group.
    return true;       // This is it! Wow. Wow. We know that you, as a bee, have
  }    // worked your whole life to get to the point where you can work for your
   // whole life. Honey begins when our valiant Pollen Jocks bring the nectar to
  // Very unhappy numbers :(                     // the hive. Our top-secret formula is
  if (q == 4  || q == 16  || q == 37 || q == 58 ||                 // automatically
      q == 89 || q == 145 || q == 42 || q == 20) {              // color-corrected,
    return false;       // scent-adjusted and bubble-contoured into this soothing
  } // sweet syrup with its distinctive golden glow you know as... Honey! - That
      // girl was hot. - She's my cousin! - She is? - Yes, we're all cousins. -
  ul sum {0};          // Right. You're right. - At Honex, we constantly strive to
  ul tmp;                // improve every aspect of bee existence. These bees are
  // stress-testing a new helmet technology. - What do you think he makes? - Not
  tmp = q%10;     // enough. Here we have our latest advancement, the Krelman. -
  sum += tmp*tmp;    // What does that do? - Oatches that little strand of honey
  q /= 10;   // that hangs after you pour it. Saves us millions. Oan anyone work
  // on the Krelman? Of course. Most bee jobs are small ones. But bees know that
```

-:--- dank.cpp<happyprime>   L28   <N>   (C++/l company WK ivy FlyC:1/0 Pro

```cpp
  q /= 10;   // that hangs after you pour it. Saves us millions. Oan anyone work
  // on the Krelman? Of course. Most bee jobs are small ones. But bees know that
```

```cpp
  tmp = q%10;      // every small job, if it's done well, means a lot. But choose
  sum += tmp*tmp;   // carefully because you'll stay in the job you pick for the
  q /= 10;     // rest of your life. The same job the rest of your life? I didn't
    // know that. What's the difference? You'll be happy to know that bees, as a
  tmp = q%10; // species, haven't had one day off in 27 million years. So you'll
  sum += tmp*tmp;    // just work us to death? We'll sure try. Wow! That blew my
  q /= 10;        // mind! "What's the difference?" How can you say that? One job
  // forever? That's an insane choice to have to make. I'm relieved. Now we only
  tmp = q%10;     // have to make one decision in life. But, Adam, how could they
  sum += tmp*tmp;   // never have told us that? Why would you question anything?
  q /= 10; // We're bees. We're the most perfectly functioning society on Earth.
  // You ever think maybe things work a little too well here? Like what? Give me
  tmp = q%10; // one example. I don't know. But you know what I'm talking about.
  sum += tmp*tmp; // Please clear the gate. Royal Nectar Force on approach. Wait
  q /= 10; // a second. Oheck it out. - Hey, those are Pollen Jocks! - Wow. I've
 // never seen them this close. They know what it's like outside the hive. Yeah,
  return is_happy(sum);  // but some don't come back. - Hey, Jocks! - Hi, Jocks!
}   // You guys did great! You're monsters! You're sky freaks! I love it! I love
     // it! - I wonder where they were. - I don't know. Their day's not planned.
          // Outside the hive, flying who knows where, doing who knows what. You
int main(int argc, char *argv[])    // can'tjust decide to be a Pollen Jock. You
{    // have to be bred for that. Right. Look. That's more pollen than you and I
  ios_base::sync_with_stdio(false);       // will see in a lifetime. It's just a
  cin.tie(NULL);     // status symbol. Bees make too much of it. Perhaps. Unless
    // you're wearing it and the ladies see you wearing it. Those ladies? Aren't
  r = generate(10000);  // they our cousins too? Distant. Distant. Look at these
  ul count; // two. - Oouple of Hive Harrys. - Let's have fun with them. It must
  cin >> count; // be dangerous being a Pollen Jock. Yeah. Once a bear pinned me
  for (ul i = 0; i < count ; ++i) {    // against a mushroom! He had a paw on my
    ul k, candidate;    // throat, and with the other, he was slapping me! - Oh,
    cin >> k >> candidate; // my! - I never thought I'd knock him out. What were
    cout << k << ' '  // you doing during this? Trying to alert the authorities.
         << candidate   // I can autograph that. A little gusty out there today,
         << (is_prime(candidate)      // wasn't it, comrades? Yeah. Gusty. We're
             && is_happy(candidate) ? " YES\n" : " NO\n");          // hitting a
  }  // sunflower patch six miles from here tomorrow. - Six miles, huh? - Barry!
    // A puddle jump for us, but maybe you're not up for it. - Maybe I am. - You
  return 0;           // are not! We're going 0900 at J-Gate. What do you think,
}      // buzzy-boy? Are you bee enough? I might be. It all depends on what 0900
```

-:--- dank.cpp<happyprime>   L70   <N>   (C++/l company WK ivy FlyC:1/0 Pro

Got this one from reddit in 2023

- try {
-     throw "the truth"; // you want the truth
- } catch (int i) { // you can't handle the truth
- }

And the “grand prize”

- Be careful, someday your irritation might get you on the front page too:
- RichardIsAFuckingIdiotControl
  - // The main problem is the BindCompany() method,
  - // which he hoped would be able to do everything. I hope he dies.
- http://mcfunley.com/from-the-annals-of-dubious-achievement

Final word

- As much as possible, let the code speak for itself
- Use the comments to tell why
- Use comments sparingly when you really need to
- Assume that one of your co-workers will one day come from a country that does not use English (and especially American idioms and pop culture)
  - So don't use pop culture in your code.