Summary:

Customers bring their computers in for repair. The problem could be with their hard drive, their RAM, their CPU, and sometimes there is no problem (PEBKAC = Problem exists between keyboard and chair).

Program:

Create a Computer each time the user says a new customer is approaching. Details of computer can either be random or set up explicitly in main through constructors. Do not make a dippy menu to create the computer. Each part of the computer has a chance to be broken (below). Try to turn the computer on, test each individual part if it doesn't turn on, offer to sell a replacement part for anything broken or missing, try to turn the computer on again.

The details of the test data doesn't matter, but it takes like 5 seconds to look up what valid values would be.

Use cases:

Here are the last few interactions at the repair counter to show what is happening.

Customer comes in with an Intel i7 @ 2.4 Ghz. It has a 1 TB SATA drive and 12 GB of RAM. After testing each part we found the CPU was broken. The customer picked an i9 3.2 Ghz to replace it

Customer comes in with Intel i9 @ 3.2Ghz. It has a 256 GB SSD and 16 GB of RAM. After testing we found nothing wrong. It turned on so they didn't buy anything

Customer comes in with a Pentium @.5GHz. It has a 2 TB SATA drive but no RAM After testing each part we found the processor and drive parts were broken. The customer picked a new item from all three categories (two broken and one missing).

After we sell the new parts, we always make sure the computer can turn on.

Data:

Computers are made of a Processor, a Drive, and RAM. A computer has to have all three parts, and have them not broken, to be able to start up. For any customer who comes in, there is a

5% chance their CPU is broken, a 15% chance their drive is broken, and a 10% chance their RAM is broken.

Class methods:

bool Computer::TurnsOn() const;

bool Computer::RAMTest() const;

bool Computer::CPUTest() const;

bool RAM::IsFunctional() const;

bool CPU::IsFunctional() const;

SP21: Yes the parts have to be dynamic. That's what replacing a part with a new one means. Yes the computer has to be dynamic. Each customer has a different computer, you aren't constantly refixing the same on. And parts don't become broken by looking at them. A part is broken or it is not. It doesn't change each time you look.