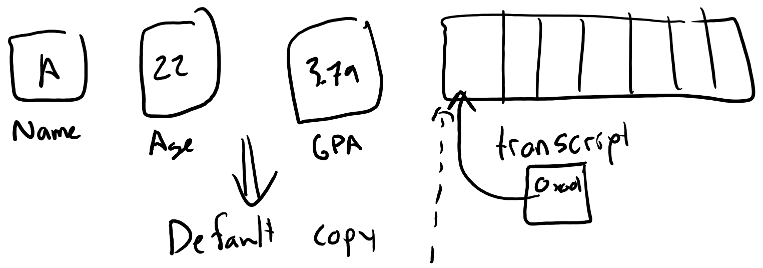
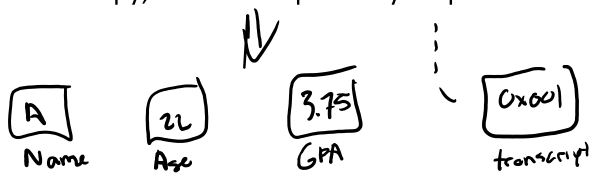
2019-11-18 CH 18 applied to Linked Lists

Wednesday, November 13, 2019 6:07 PM

- 18.1: What does it mean to copy something?
  - Colloquial conceptualization of copy -> Make an exact replica of an item
  - Also somewhat straightforward for basic variables (int, char, double)
    - 5 => 5
  - Less clear on complex data. Consider a "student" class
    - Name
    - Age
    - GPA
    - Transcript (list of prior courses)

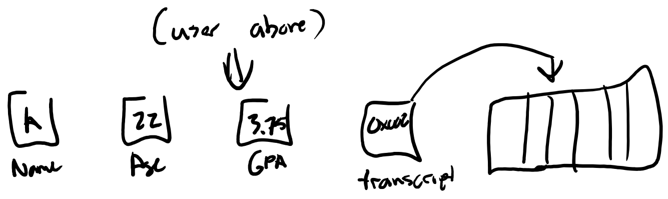


- In a default copy, values are "perfectly" copied



- In programming terms, the default copy is called a "shallow copy"
  - 18.3.3 Shallow vs deep copy

- In a deep copy, all data is replicated for a copy

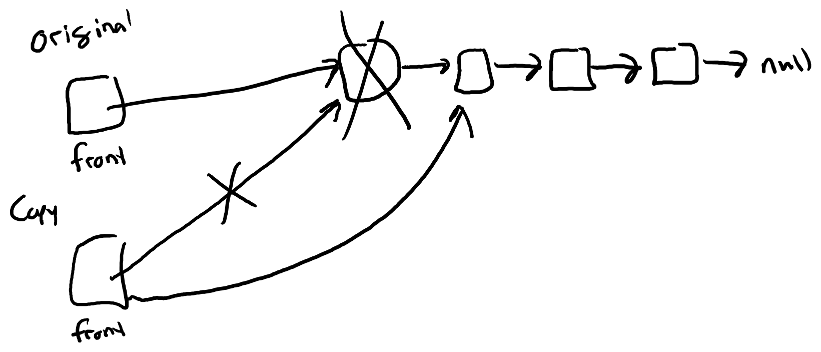


- Deep copies preserve encapsulation of data, but are slow because **everything** must be replicated (e.g. students have transcripts which have courses which have other students which have more transcripts)

- Example code:

```
Vector<int> data = populate_data(); //fill vector with stuff
Vector<int> copy_data = data; //is this a shallow copy?
Copy_data[0] = 5;
```

Why did our example LL code break the original?



Why might we want shallow or deep copy?

- Shallow copies are very fast
- Shallow copies are also very helpful when you want to write a function that modifies an input parameter
    - Sort of the idea behind pass by reference / pass by pointer
- Deep copies are necessary when you want a complete, exact replica of data
    - But they're slow
    - Deep copies are often necessary when working with data structures or objects with dynamically allocated memory

- 18.3.1 Copy constructors
- 18.3.2 Copy assignment
- 18.3.4 Move operations
- 18.4 Essential operations for classes that use dynamic memory
    - Constructors from one or more arguments
    - Default constructor
    - Copy constructor (copy object of same type)
    - Copy assignment (copy object of same type)
    - Move constructor (move object of same type)
    - Move assignment (move object of same type)
    - Destructor