Example use of IntList

```cpp
void print(IntList il) {
    while (!il.empty())
        cout << il.pop() << " ";
    cout << endl;
}
```

Sample main

```cpp
int main(int argc, char ** argv) {
    IntList s1;
    for(int i = 0; i < 3; i++)  s1.push(i);
    print(s1); // 2 1 0
    IntList s2(s1);
    for(int i = 3; i < 6; i++)s2.push(i);
    print(s2); // 5 4 3 2 1 0
    s1.appendCopyAtEnd(s2);
    print(s1); // 2 1 0 5 4 3 2 1 0
    print(s2); // 5 4 3 2 1 0
    for(int i = 0; i < 6; i++)
        s2[i] = 2*i;
    print(s1); // 2 1 0 5 4 3 2 1 0
    print(s2); // 0 2 4 6 8 10
}
```

Things to watch for

- `s1.appendCopyAtEnd(s1)`
- `s1 = s1;`
**Shared Rep, Copy on write**

- Consider
  ```cpp
def f(const IntEArray &a) {
    IntEArray b(a);
    // at this point, a and b should
    // share a representation
    b[5] = 42;
    // at this point, they don’t
    ...
  }
```  

**Shared representation**

- Each object has pointer to representation
- Representation has reference count
- If ref count > 1, can’t modify representation
- But can copy it

**Reference count maintenance**

- Need to update reference count in things like copy constructor, destructor, …
  - delete representation if ref count goes to zero

**Expandable**

- Referencing an element that hasn’t been allocated yet
  - allocates a new, larger representation
  - copies data from old representation
  - new size should be at least 1.5x larger than the old one
    - for some value of 1.5

**ElementRef**

- Add
  ```cpp
  int get(int index);
  void set(int index, int value)
  ```
  functions to IntEArray
- Write ElementRef entirely in terms of get and set
  - doesn’t need any other functionality

**startingFrom(int offset)**

- Returns a new IntEArray
  - index 0 of new array same as index offset of old array
- Should share representation