PIC10B: Final Review

Weiqi Chu
UCLA Mathematics

PIC 10B – Final review

- Pointers
- Inheritance
- Streams
- Recursion
- Sorting and Searching
- Linear Data Structures
- Trees
- Operator Overloading

Operator Overloading Summary

- Constraints of overloading operators
  - Existing operators, one class type operand, etc.
- Member/Non-member functions
- Arithmetic Operators
  - +, -, *, /, -
- Comparison Operators
  - <, <=, >, >=, ==
- Input/Output Operators
  - <<, >>
- Conversion operator
  - int -> Fraction
  - Fraction -> double

Operator Overloading Summary

Example: Any Error? Or what's the output?

```
Employee operator*(Employee e, double percent)
{
    double new_salary = e.get_salary()*(1+percent);
    e.set_salary(new_salary);
    Employee new_e = e;
    return new_e;
}

Employee tom("Tom", 2000);
Employee jim("Jim", 1000);
jim = tom*0.5;
cout << jim.get_name() << " " << jim.get_salary()<< endl;
cout << tom.get_name() << " " << tom.get_salary()<< endl;
```

Ans:

Tom 3000

Tom 2000

Operator Overloading Summary

Example: Any Error? Or what's the output?

```
Employee operator*(Employee e, double percent)
{
    double new_salary = e.get_salary()*(1+percent);
    e.set_salary(new_salary);
    Employee new_e = e;
    return new_e;
}

Employee tom("Tom", 2000);
Employee jim("Jim", 1000);
jim = 0.5*tom;
cout << jim.get_name() << " " << jim.get_salary()<< endl;
cout << tom.get_name() << " " << tom.get_salary()<< endl;
```

Ans:

jim = 0.5*tom; has error.

No conversion from Employee to double.

Operator Overloading Summary

Example: Rewrite `operator*` as a member function of Employee, such that the following code has the same output.

```
Employee operator*(Employee e, double percent)
{
    double new_salary = e.get_salary()*(1+percent);
    e.set_salary(new_salary);
    Employee new_e = e;
    return new_e;
}
Employee tom("Tom", 2000);
Employee jim("Jim", 1000);
jim = tom*0.5;
cout << jim.get_name() << " " << jim.get_salary()<< endl;
cout << tom.get_name() << " " << tom.get_salary()<< endl;
```

Ans:

```
Employee Employee::operator*(double percent)
{
    double new_salary = salary*(1+percent);
    Employee new_e(name, new_salary);
    return new_e;
}
```

Trees

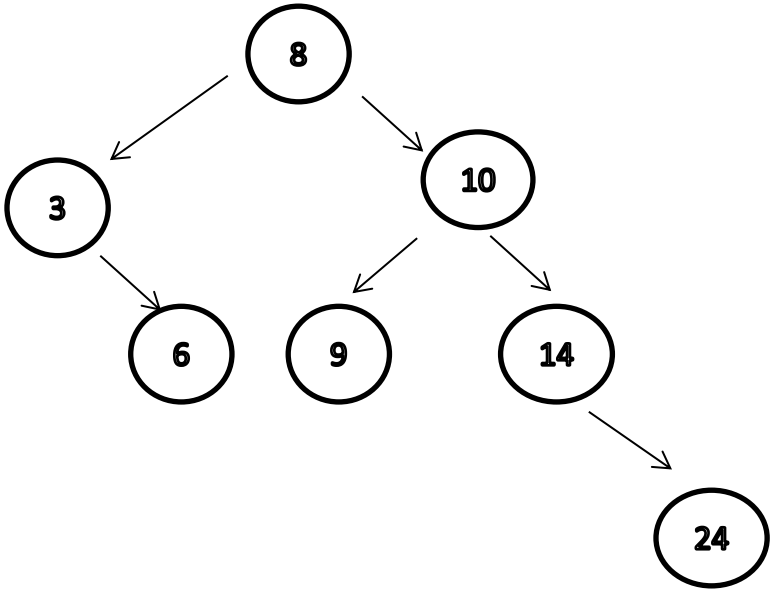
- STL classes: set, map, unordered_map, priority_queue
- Binary Search Tree implementation
  - insert, erase, smallest, traversal, print etc.
- Heap implementation
  - Insert, erase
  - Memory storage -- array
- Complexity of operations in Big-Oh notation

Trees

- Example: Draw a diagram of a binary search tree, with elements inserted as following

8, 10, 14, 3, 9, 6, 24

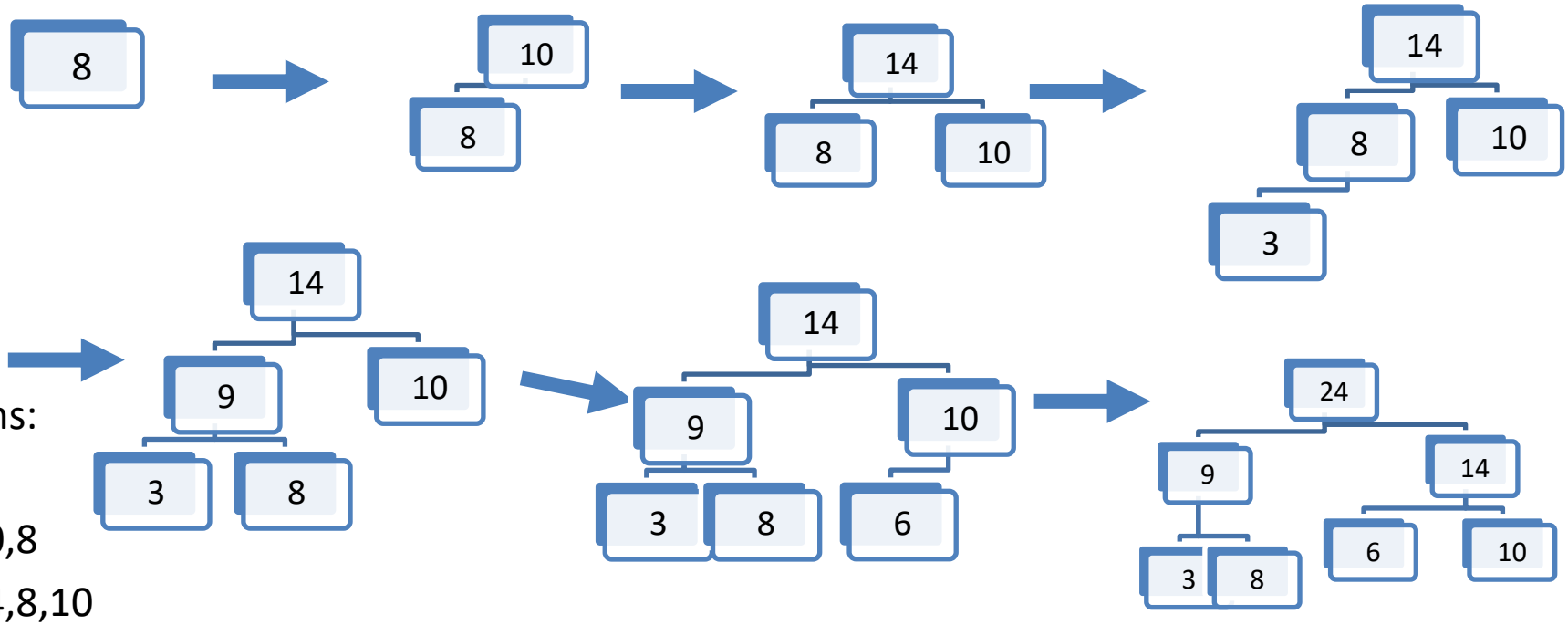
- Answer:



Trees

- Example: Add the following elements to a heap. Use arrays to illustrate the heap structure at each step.

8, 10, 14, 3, 9, 6, 24



Ans:

8

10,8

14,8,10

14,8,10,3

14,9,10,3,8

14,9,10,3,8,6

24,9,14,3,8,6,10

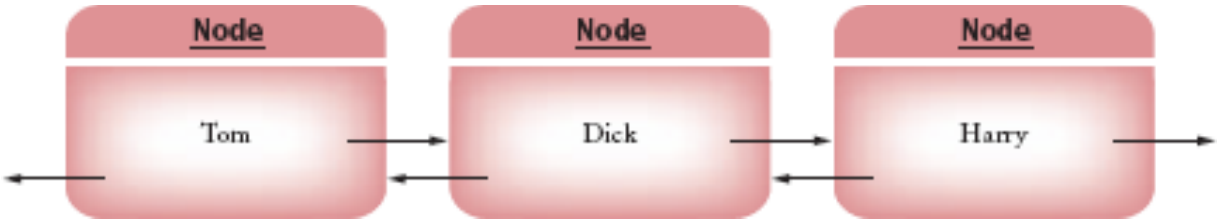
Trees

- Example: Consider a set of distinct data without ordering relation. Which data structure in C++ STL could be used to store it?
    - Sets
    - Priority Queues
    - Maps
    - Vectors
    - Binary Search Trees

- Answer: Vectors

Linear Data Structures

- Linked List implementation
  - Node, Iterator, Linkedlist classes
  - Methods



- STL classes
  - List
  - Queue, FIFO (First in, First Out)
  - Stack, LIFO (Last In, First Out)

Linear Data Structures

- Examples: Find the error

```
#include <list>
int main()
{
    std::list<int> IntList { 1, 23, 0 };
    std::cout << IntList[2] << std::endl;
    return 0;
}
```

Answer:

cannot access IntList[2].

Linear Data Structures

Examples: What is the output of the following code

```
#include <stack>;
 . . .
stack<string> s;
s.push("Tom");
s.push("Jerry");
s.push("Spike");
while (s.size() > 0)
{
  cout << s.top()<< " ";
  s.pop();
}
```

Answer:

Spike Jerry Tom

Linear Data Structures

- Priority queues are not queues!
- Examples: What is the output of the following code

```
#include <queue>;
  . . .
priority_queue<string> s;
s.push_back("Tom");
s.push_back("Jerry");
s.push_back("Spike");
while (s.size() > 0)
{cout << s.front () << " ";
s.pop_back();}
```

Answer:

Tom Spike Jerry

Lecture 27: Final Review

Weiqi Chu
UCLA Mathematics

Sorting and Searching

- Big-Oh notation
- Understand/implement the following algorithms
  - Selection Sort : O (n^2)
  - Merge Sort : O (n log (n))
  - Linear Search : O (n)
  - Binary Search : O (log n)
- Design algorithms for related questions
  - Search the largest element in a vector
  - Search the second largest element in a binary search tree

Sorting and Searching

Examples:

Which type of algorithm will be executed in the following code snippet?

```
int myfunc( int arr[], int n, int a)
{
    for (int i = 0; i < n; i++)
    {
        if (arr[i] == a)
        return i;
    }
    return -1;
}
```

Answer:

A linear search

Sorting and Searching

Examples:

n is the size of array a. What is the order of complexity of the following function?

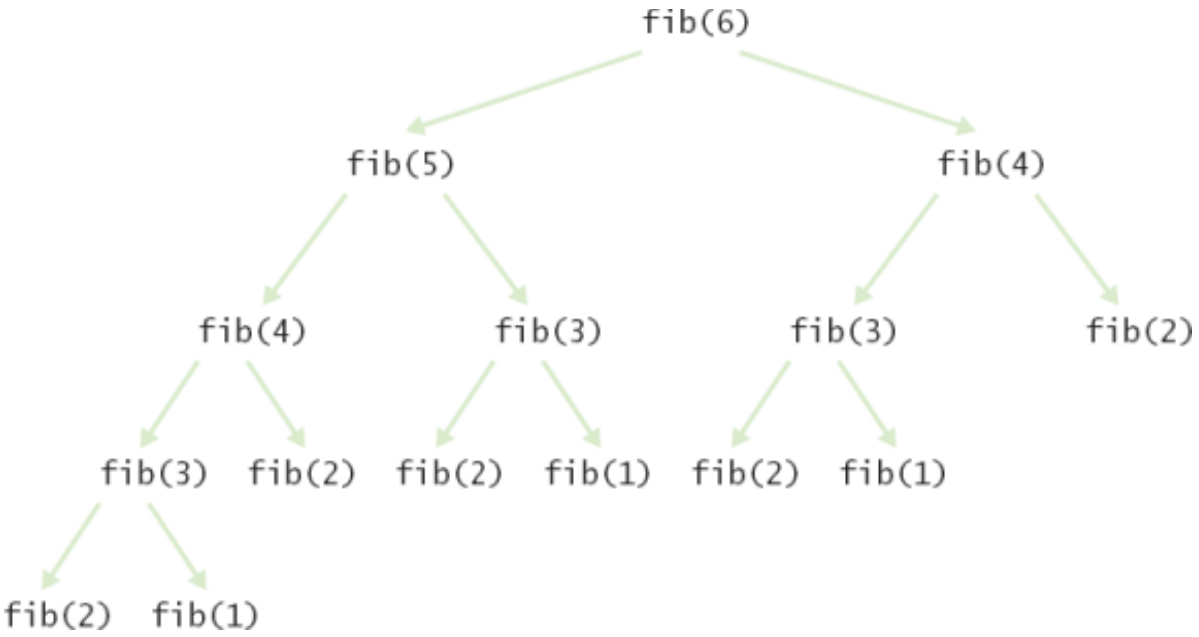
```
int mystery(int a[], int n)
{
   int i; int j;
   int t = 0;
   for (int i = 0; i < n; i++)
   {
      for (int j = 0; j < n; j++)
      {
         t = t + a[i] * a[j];
      }
   }
   return t;
}
```

Answer:

O (n^2)

Recursion

- Recursion: initial values (special cases) + recursive relation
- Common Error: Infinite Recursion
- Examples of recursion
  - Permutation, Tower of Hanoi, Palindrome, Fibonacci, etc
- Complexity analysis of recursion



Recursion

Example: What is the output of the following code snippet?

```
int myfunction(int n)
{return n * myfunction(n - 1);}
int main()
{
  cout << myfunction(3) << endl;
  return 0;
}
```

Answer: Infinite recursion

How to fix it? Add a stopping condition

```
int myfunction(int n)
{     if (n < 2)
          { return 1; }
      return n * myfunction(n - 1);
}
```

Recursion

Examples : What is myfunction(3)? What does following function do?

```
int myfunction(int n)
{
   if (n < 2) { return 1; }
   return n * myfunction(n - 2);
}

int main()
{cout << myfunction(3) << endl;
 return 0;}
```

Answer:
myfunction(3) = 3

The function computes the double factorial or semi-factorial of n

n!! = n * (n-2) * (n-4) * ...

5!! = 5 * 3 * 1;

8!! = 8 * 6 * … * 2;

Recursion

Exercise:

n is the size of array a. Rewrite this function using recursion, so that they have the same functionality.

```
int mystery(int a[], int n)
{
   int i; int j;
   int t = 0;
   for (int i = 0; i < n; i++)
   {
      for (int j = 0; j < n; j++)
      {
         t = t + a[i] * a[j];
      }
   }
   return t;
}
```

Recursion

Exercise:

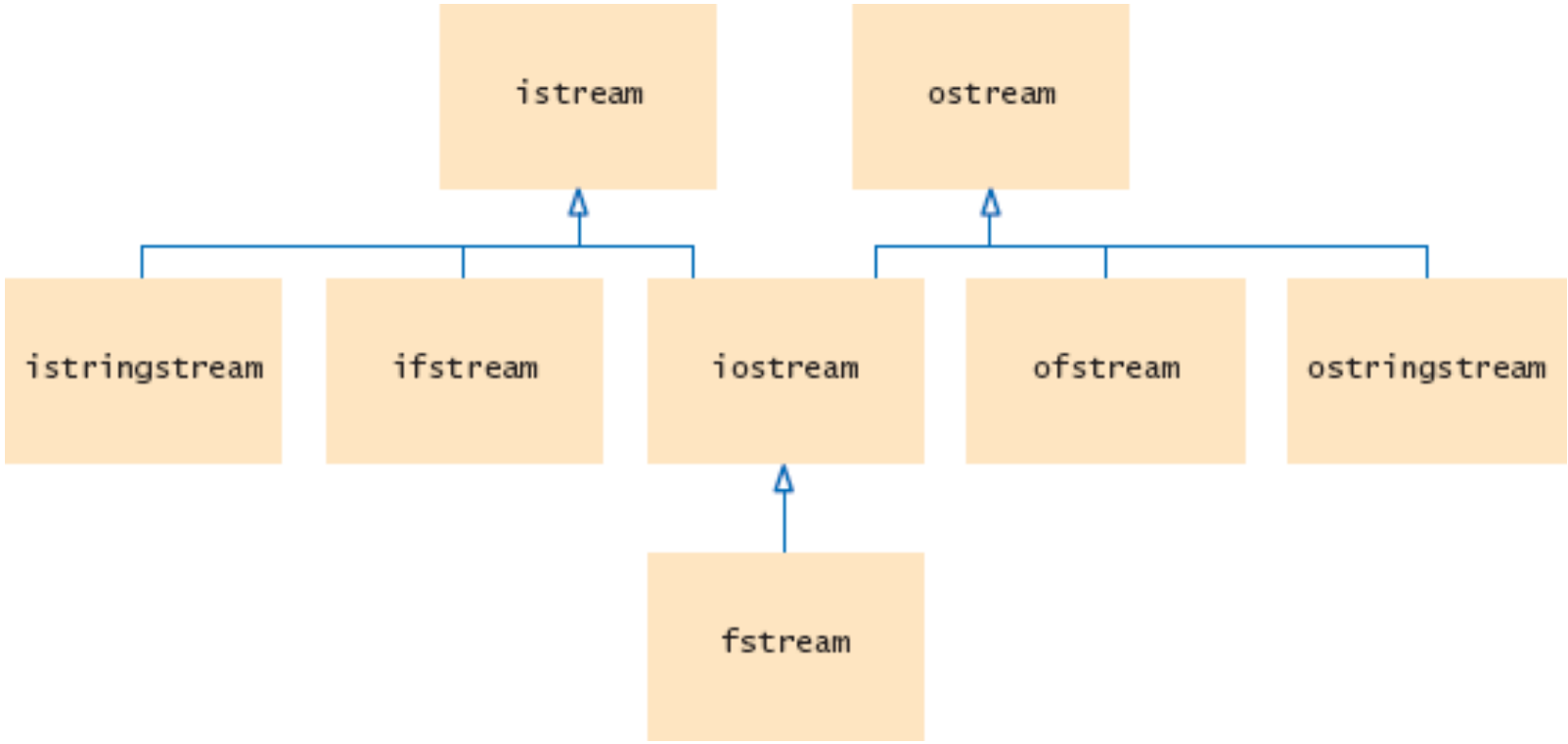
n is the size of array a. Rewrite this function using recursion, so that they have the same functionality.

mystery(a, n) = a[0]*a[0] + a[0]*a[1] + … + a[0]*a[n-1]
+ a[1]*a[0] + a[1]*a[1] + … + a[1]*a[n-1]
…
+ a[n-1]*a[0] + a[n-1]*a[1] + … + a[n-1]*a[n-1]

```
int mystery(int a[], int n)
{
   if (n<=0)
       return 0;
   else
   {
       int s = mystery(a, n-1);
       for (int i = 0; i < n-1; i++)
          s += 2*a[n-1]*a[i];
       s += a[n-1]*a[n-1];
       return s;
   }
}
```

Streams Summary

- `<fstream>` Read from or Write to files
- `<sstream>` Read from or Write to strings
- `<iomanip>` Formatted output
- Member function
  - get(), unget(), fail(), etc
- Random Access



Streams Summary

Example: What does the following function do?

```
double read (ifstream& in)
{
  double first, second;
  if (in >> first) ;
  else
      return -100;
  while (in >> second)
  {
   if (first == second)
        return first;
   else
        first = second;
  }
  return -100;
}
```

Answer: To exam if any two successive numbers are equal. If found, return the number, otherwise return -100.

Streams Summary
Exercise: File I/O

Write a full program starting from #include that accomplishes the following task.

- First, open the file myFile.txt for reading.
- Next, write the contents of the file in reverse order into a file called backwards.txt.
- For example, if the file contains the phrase:

```
Hello! How's it going?
```

- The file backwards.txt should have the content

```
going? it How's Hello!
```

```cpp
#include<iostream>
#include<fstream>
#include<string>
#include<vector>
using namespace std;

int main ()
{
    // construct an input file stream object
    ifstream fin;
    fin.open("myFile.txt");
    // deal with error
    if (fin.fail())
    {
        cout << "Error!\n";
        return 1;
    }
    // construct an output stream
    ofstream fout;
    fout.open("backwards.txt");
    // deal with error
    if (fout.fail())
    {
        cout << "Error!\n";
        return 1;
    }
    // use a vector container to store words info
    string word;  // store a word temperarily
    vector<string> myfile; // store all words as elements
    while (fin >> word) // traversal of the file and store everything in the vector
        myfile.push_back(word);
    for (int i=myfile.size()-1; i>=0; --i) // output words to file
        fout << myfile[i] << " ";
    fin.close();
    fout.close();
    return 0;
}
```

I highly suggest you writing on your own before checking on the solutions!

Inheritance

- **base class** — the more general class
- **derived class** — the more specialized class, inherits from the base class

- Polymorphism:
  (1) the same function name in base and derived classes
  (2) base class pointer to call that function
  (3) **virtual** keyword to the base class function

- Dynamic binding vs Static binding

Example:

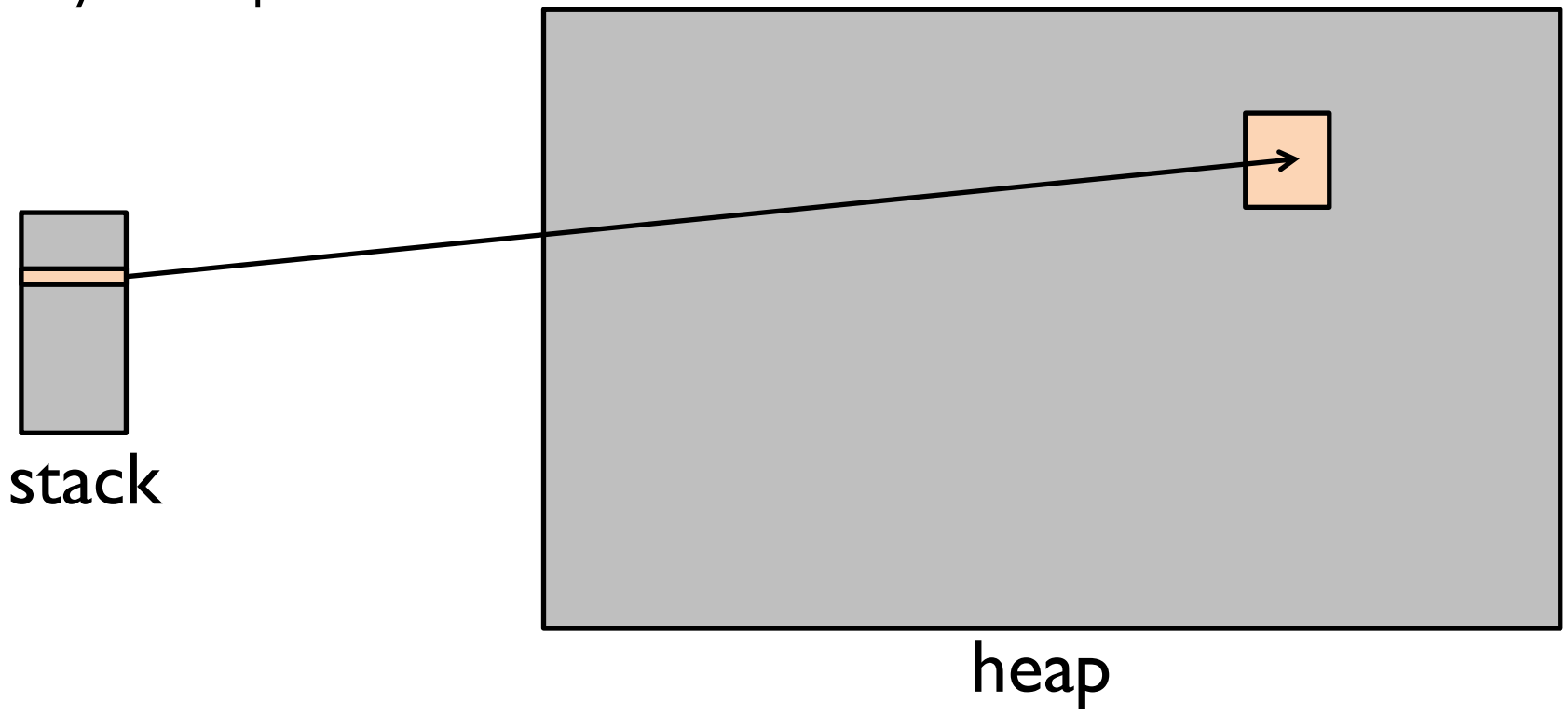
Any error?
What's the output?

Answer:
Total area: 35

```
 5 // Base class
 6 class Shape{
 7     public:
 8         //....
 9         void setWidth(int w){width=w;};
10         void setHeight(int h){height=h;};
11     protected:
12         int width;
13         int height;
14 };
15 // Derived class
16 class Rectangle: public Shape
17 {  public:
18     int getArea() { return (width * height); }
19     //....
20 };
21 int main()
22 {
23     Rectangle Rect;
24     Rectangle *PRect=&Rect;
25     Rect.setWidth(5);
26     Rect.setHeight(7);
27     // Print the area of the object.
28     cout << "Total area: " << PRect->getArea() << endl;
29     return 0;
30 }
```

Pointers

- Declaring a pointer Employee * p = new Employee();
- Stack/Heap variables
- Avoid **Memory Leaks**
- Dereference pointers
- Arrays and pointers



Pointers

```
32 int main()
33 {
34     int* p1 = new int(10);
35     int* p2;
36     int n = 100;
37     p2 = &n;
38     cout << "Value stored at pointer p1 is: " << *p1 << endl;
39     cout << "Value stored at pointer p2 is: " << *p2 << endl;
40     delete p1;
41     delete p2;
42     return 0;
43 }
```

Questions:

(1) Any ERROR?

Line 35: int*p2=NULL;

Line 41: unable to reclaim space on the stack

(2) Stack or Heap variables?

Stack variables: p1, p2, n

(3) After fixing errors, what's the output of the first line?

a) Value stored at pointer p1 is: 10

b) Value stored at pointer p1 is: 100

c) Value stored at pointer p1 is: <an unpredictable value>

Answer: a.

Pointers

Example: What is the output of the following code?

```
1  char* my_cut(char x[])
2  {
3      int num_chars = 0;
4      while (x[num_chars] != '\0')
5          num_chars++;
6      x[num_chars / 2] = '\0';
7      return x;
8  }
9  int main()
10 {
11     char s[] = "ABCDE";
12     cout << my_cut(s);
13     return 0;
14 }
```

Output:

```
AB
```

Thank you!