CSE 100: BSTS AND C++

```cpp
#include <iostream>
using namespace std;

int main()
    {cout<<"Hola Facebook\n";
     return 0;
    }
```
Announcements

• PA 1 released due next Wed in the morning at 10am
• Discussion section today (Section B)
  • Topic: Getting started with PA1 (Section A watch podcast)
• Reading Quizzes
  • Zybook activities count as quizzes.
  • You must complete the Zybook activities 8.1.1 to 8.3.2 and 14.3.1 to 14.7.2 (4/3) at 11:59 pm to receive credit
  • Your progression through these sections is indicated in the book and you can track yourself.
Goals for today

• Draw memory model diagrams for C++ pointers and references
• Explain C++ code for implementing binary search trees
• Explain pass-by-reference and constants in C++
Integrity Guidelines

Basic rules

• Do not look at or copy other people’s code and do not share your code with others (other than your partner). Period.
• “Other people” includes what you can find/share on the internet.
• Read the Integrity Statement carefully. Ask if you have questions.

Integrity

• You will be tested on your ability to understand and write code for data structures in this class (and invariably during interviews)
• Cheaters will likely get “caught” during the exam because exams, for the most part, make your grade in this class.
• Why else shouldn’t you cheat?
  • Its unethical
  • Its unfair to students who do the work legitimately
  • Hurts the reputation of the UCSD CSE degree
BST Operations

• Your first Programming Assignment will ask you to implement:
  • Find
  • Insert
  • Clear
  • Size
  • An Iterator
  • (A few other methods)

• We will assume that you have already seen these operations and/or can learn them from the reading. We will not explicitly cover (most of) them in class.
Under the hood: Finding the successor of an element in the BST

Which node is the successor of 56? How would you find it?
Today’s topic: C++

C++’s main priority is getting correct programs to run as fast as it can; incorrect programs are on their own.

Java’s main priority is not allowing incorrect programs to run; hopefully correct programs run reasonably fast, and the language makes it easier to generate correct programs by restricting some bad programming constructs.

-- Mark Allen Weiss, C++ for Java Programmers

Why C++ for data structures?
In Java:

```java
public class BSTNode {
    public BSTNode left;
    public BSTNode right;
    public BSTNode parent;
    public int data;

    public BSTNode( int d ) {
        data = d;
    }
}
```

C++, attempt 1:

```cpp
class BSTNode {
    public BSTNode left;
    public BSTNode right;
    public BSTNode parent;
    public int data;

    public BSTNode( const int & d ) {
        data = d;
    }
};
```

Which of the following is a problem with the C++ implementation above?
A. You should not declare the types of your variables in C++
B. The class BSTNode should be declared public
C. The semicolon at the end of the class will cause a compile error
D. In C++ you specify public and private in regions, not on each variable or function
In Java:

```java
public class BSTNode {
    public BSTNode left;
    public BSTNode right;
    public BSTNode parent;
    public int data;

    public BSTNode( int d ) {
        data = d;
    }
}
```

C++, attempt 2:

```cpp
class BSTNode {
public:
    BSTNode* left;
    BSTNode* right;
    BSTNode* parent;
    int data;

    public BSTNode( const int & d ) {
        data = d;
    }
};
```

- The other problem is with how we have declared left, right and parent above.
- They should be `BSTNode*` (pointers to `BSTNodes`) and not `BSTNode` type.
Which of the following statements is true about this code?

```
int a = 5;
int b = a;
int* pt1 = a;
```

A. Both pt1 and b can be used to change the value of a.
B. Only pt1 can be used to change the value of a.
C. This code causes a compile error.
Pointers in C++

```cpp
int a = 5;
int b = a;
int* pt1 = &a;
```

<table>
<thead>
<tr>
<th>address</th>
<th>memory cell</th>
<th>identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>512000</td>
<td>5</td>
<td>a</td>
</tr>
<tr>
<td>512004</td>
<td>5</td>
<td>b</td>
</tr>
<tr>
<td>512008</td>
<td>512000</td>
<td>pt1</td>
</tr>
</tbody>
</table>
Pointers in C++

```cpp
int a = 5;
int b = a;
int* pt1 = &a;

pt1 = &b;
```

```
a: 5
b: 5
pt1: 
```
Pointers in C++

class MyClass {
    private:
        int a;
    public:
        void setA(int a) { this->a = a; }
        int getA() { return a; }
};

What will the line
MyClass c;
do?
A. Declare a variable of type MyClass, but not create an object
B. Declare a variable of type MyClass and create an object of type MyClass
C. Declare a variable of type pointer to MyClass, but not create an object
D. Declare a variable of type pointer to MyClass, and create an object of type MyClass
Pointers in C++

class C {
    private:
        int a;
    public:
        void setA(int a) { this->a = a; }
        int getA() { return a; }
};

What will the line
C* c;
do?
A. Declare a variable of type C, but not create an object
B. Declare a variable of type C and create an object of type C
C. Declare a variable of type pointer to C, but not create an object
D. Declare a variable of type pointer to C, and create an object of type C
int main() {
    C* x; // declare x to be a pointer to a C object
    x = new C(); // create a C object, and make x point to it
    x->setA(5); // dereference x, and access a member
    // note: (*x).setA(5) is equivalent
}

x:  
C  
   a: setA(5)

a: 5
Pointers in C++

```cpp
int main() {
    C* x; // declare x to be a pointer to a C object
    x = new C(); // create a C object, and make x point to it
    x->setA(5); // dereference x, and access a member
    // note: (*x).setA(5) is equivalent
    C* y = x;
}
```

Which represents the new diagram?

A.  

B.  

C.  

D. The line in red causes an error
int main() {
    int d = 5;
    int & e = d;
}

The diagram that represents the code above is C

D. This code causes an error
int main() {
    int d = 5;
    int & e = d;
    int f = 10;
    e = f;
}

How does the diagram change with this code?

A. d: 10  e: 10  f: 10
B. d: 5  e: 10  f: 10
C. d: 10  e: 10  f: 10
D. Other or error
Pointers and references. Draw the picture for this code is the one on the left.

```c
int a = 5;
int & b = a;
int* pt1 = &a;
```

What are three ways to change the value in the box to 42?
Pointers and references. Draw the picture for this code is the one on the left.

```cpp
int a = 5;
int & b = a;
int* pt1 = &a;
```
int main() {
    int const d = 5;
    int & e = d;
}

Does this code have an error? If so, why?

A. No, there is no error
B. Yes, there is an error because ints cannot be constant in C++
C. Yes, there is an error because a reference to a constant must also be declared constant
int main() {
    int const d = 5;
    const int & e = d;
}
C++, attempt 3:

```cpp
class BSTNode {
public:
    BSTNode* left; // left pointer
    BSTNode* right; // right pointer
    BSTNode* parent; // parent pointer
    int const data; // data

    BSTNode( const int & d ) {
        // body here
    }
};
```

The code above in red specifies that \( d \) is passed by constant reference.

Imagine this code works (it doesn’t yet). If it did, consider creating a new `BSTNode` as follows:

```cpp
int myInt = 42;
BSTNode* myNode = new BSTNode( myInt );
```

Which of the following diagrams best represents what that means?

A. [Diagram A]

B. [Diagram B]

C. [Diagram C]

D. This code has an error

A. MyInt: 42  
   d: 42  
   d is not allowed to change what's in its box

B. MyInt: 42  
   d: (empty)  
   The address in d's box can't be changed

C. MyInt: 42  
   d: 42  
   d can't change myInt because there's no connection

D. This code has an error
In Java:

```java
public class BSTNode {
    public BSTNode left;
    public BSTNode right;
    public BSTNode parent;
    public int data;

    public BSTNode( int d ) {
        data = d;
    }
}
```

C++, attempt 3:

```cpp
class BSTNode {
public:
    BSTNode* left;
    BSTNode* right;
    BSTNode* parent;
    int const data;

    BSTNode( const int & d ) {
        data = d;
    }
};
```

Another problem with the C++ implementation above is that:

Because data is a constant variable, the constructor will cause an error.
In Java:

```java
public class BSTNode {
    public BSTNode left;
    public BSTNode right;
    public BSTNode parent;
    public int data;

    public BSTNode( int d ) {
        data = d;
    }
}
```

One more potential issue....

C++, attempt 4:

```cpp
class BSTNode {
public:
    BSTNode* left;
    BSTNode* right;
    BSTNode* parent;
    int const data;

    BSTNode( const int & d ) :
    data(d) {
    }
}
```
In Java:

```java
public class BSTNode {
    public BSTNode left;
    public BSTNode right;
    public BSTNode parent;
    public int data;

    public BSTNode( int d ) {
        data = d;
    }
}
```

C++, attempt 5:

```cpp
class BSTNode {
public:
    BSTNode* left;
    BSTNode* right;
    BSTNode* parent;
    int const data;

    BSTNode( const int & d ) :
    data(d) {
        left = right = parent = 0;
    }
};
```

ALWAYS initialize in C++. C++ won’t do it for you. Why not?