Pointers and References

A variable containing an instance of an object:

```c
Sphere s1;
```

A reference variable of a Sphere object:

```c
Sphere & s1;
```

A variable containing a pointer to a Sphere object:

```c
Sphere * s1;
```
Pointers

Three key ideas:

1.

2.

3.
```cpp
#include <iostream>
#include "sphere.h"

int main() {
    cs225::Sphere s;
    std::cout << "Address storing `s`:" << &s << std::endl;

    cs225::Sphere *ptr = &s;
    std::cout << "Addr. storing ptr: " << &ptr << std::endl;
    std::cout << "Contents of ptr: " << ptr << std::endl;

    return 0;
}
```
Indirection Operators

Given any variable $v$:

$\&v$

$*v$

$v->$
Stack Memory
```cpp
int main() {
    int a;
    int b = -3;
    int c = 12345;
    int *p = &b;
    return 0;
}
```
```cpp
#include <iostream>

int main() {
    std::cout << sizeof(int) << std::endl;
    return 0;
}
```
```cpp
#include <iostream>

int main() {
    std::cout << sizeof(int *) << std::endl;
    return 0;
}
```
int main() {
    int a;
    int b = -3;
    int c = 12345;
    int *p = &b;
    return 0;
}
```cpp
#include "sphere.h"

int main() {
    cs225::Sphere s;
    cs225::Sphere *p = &s;
    return 0;
}
```
#include <iostream>
#include "sphere.h"

int main() {
    std::cout << sizeof(cs225::Sphere) << std::endl;
    std::cout << sizeof(cs225::Sphere *) << std::endl;
    return 0;
}
```c
int hello() {
    int a = 100;
    return a;
}

int main() {
    int a;
    int b = -3;
    int c = hello();
    int d = 42;
    return 0;
}
```
Problems of the Day (POTD)

**POTDs** are small, daily problems for you to practice programming in an environment similar to the CBTF exam environment.

Each POTD is worth +1 extra credit point, capped at +40. *(Course-wide, all extra credit is capped at +100.)*

_POTD#1 is available now, until 8:00am tomorrow morning when POTD#2 becomes available!_
#include "sphere.h"

using namespace cs225;

Sphere *CreateUnitSphere() {
    Sphere s(1);
    return &s;
}

int main() {
    Sphere *s = CreateUnitSphere();
    double r = s->getRadius();
    double v = s->getVolume();
    return 0;
}
#include "sphere.h"

using namespace cs225;

Sphere *CreateUnitSphere() {
    Sphere s(1);
    return &s;
}

int main() {
    Sphere *s = CreateUnitSphere();
    double r = s->getRadius();
    double v = s->getVolume();
    return 0;
}
#include "sphere.h"

using namespace cs225;

Sphere *CreateUnitSphere() {
    Sphere s(1);
    return &s;
}

int main() {
    Sphere *s = CreateUnitSphere();
    double r = s->getRadius();
    double v = s->getVolume();
    return 0;
}
What happens on a real system?
int main() {
    Sphere *s = CreateUnitSphere();
    cout << s->getRadius() << endl;
    cout << "s->getRadius(): " << s->getRadius() << endl;
    cout << "&s (main): " << &s << endl;
    cout << " s (main): " << s << endl;
    double r = s->getRadius();
    cout << "&r (main): " << &r << endl;
    cout << " r (main): " << r << endl;
    double v = s->getVolume();
    cout << "&v (main): " << &v << endl;
    cout << " v (main): " << v << endl;
}

Real results when running on linus.ews.illinois.edu
&s (CreateUnitSphere): 0x7ffee6bf5ca8
s (main): 0x7ffee6bf5ca8
s->getRadius(): 2.07941e-317
&s (main): 0x7ffee6bf5d30
s (main): 0x7ffee6bf5ca8
&r (main): 0x7ffee6bf5d28
r (main): 6.95312e-310
&v (main): 0x7ffee6bf5d20
v (main): 0
Stack Memory vs. Heap Memory

Stack Memory

Heap Memory

0xf0

0x42000

0x42008

0x42010

0x42018

0x42020

0x42028

0x42030

0x42038

0x42040

0x42048
Heap Memory - new

As programmers, we can use heap memory in cases where the lifecycle of the variable exceeds the lifecycle of the function.

The only way to create heap memory is with the use of the **new** keyword. Using **new** will:

1.

2.

3.
Heap Memory - delete

2. The only way to free heap memory is with the use of the `delete` keyword. Using `delete` will:
   - 
   - 

3. Memory is never automatically reclaimed, even if it goes out of scope. Any memory lost, but not freed, is considered to be “leaked memory”.
```cpp
#include "sphere.h"
using namespace cs225;

int main() {
    int *p = new int;
    Sphere *s = new Sphere(10);
    return 0;
}
```
```cpp
#include "sphere.h"
using namespace cs225;

int main() {
    Sphere *s1 = new Sphere();
    Sphere *s2 = s1;
    s2->setRadius(10);
    return 0;
}
```
```cpp
#include <iostream>
using namespace std;

int main() {
    int *p;
    int x;

    p = &x;
    x = 6;

    cout << x << endl;
    cout << p << endl;

    return 0;
}
```
```cpp
#include <iostream>
using namespace std;

int main() {
    int *p, *q;
    p = new int;
    q = p;
    *q = 8;
    cout << *p << endl;

    q = new int;
    *q = 9;
    cout << *p << endl;
    cout << *q << endl;
    return 0;
}
```
CS 225 – Things To Be Doing

**Exam 0** starts tomorrow
Ensure you have signed up for your Exam 0 timeslot!

**MP1** is available now!
Due: Monday, Jan 29th *(one week from today)*

**POTD**s released every day
+1 extra credit /completed POTD

**Office Hours,** every day except Tuesday
Check the “calendar” link on the CS 225 website for more details