CS 225

Data Structures

Wade Fagen-Ulmschneider

Pointers and References

A variable containing an instance of an object:

```
1 Sphere s1;
```

A reference variable of a Sphere object:

```
1 Sphere & s1;
```

A variable containing a pointer to a Sphere object:

```
1 Sphere * s1;
```

Pointers

Three key ideas:

1.

2.

3.

main.cpp

```
 1 #include <iostream>
 2 #include "sphere.h"
 3
 4 int main() {
 5   cs225::Sphere s;
 6   std::cout << "Address storing `s`:" << &s << std::endl;
 7
 8   cs225::Sphere *ptr = &s;
 9   std::cout << "Addr. storing ptr: "<< &ptr << std::endl;
10   std::cout << "Contents of ptr: "<< ptr << std::endl;
11
12   return 0;
13 }
```

Indirection Operators

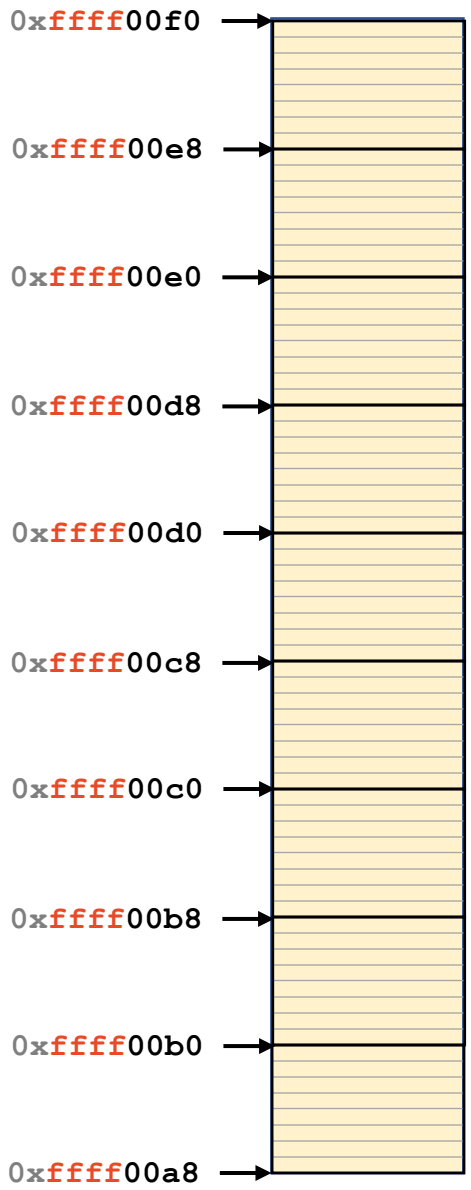
Given any variable v:

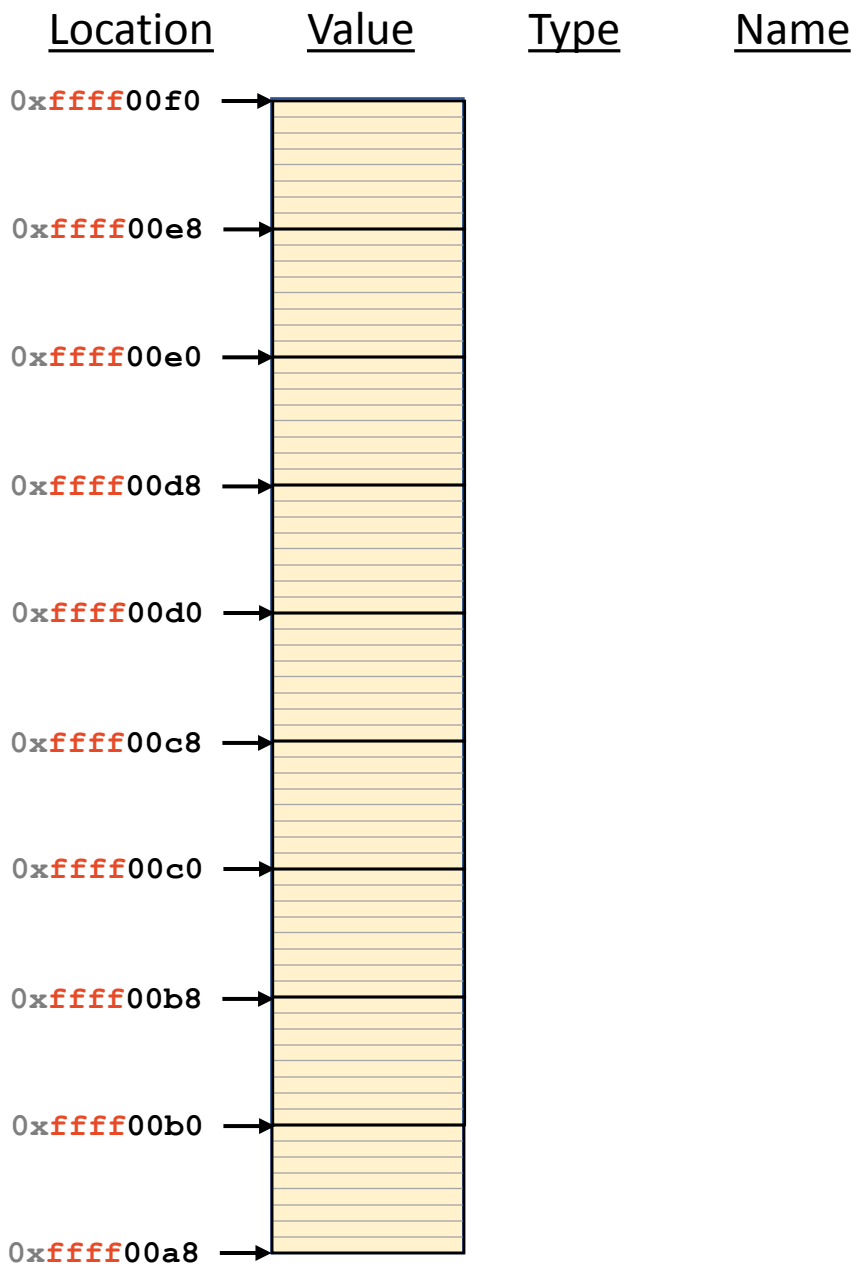
&v

***v**

v->

Stack Memory





example1.cpp

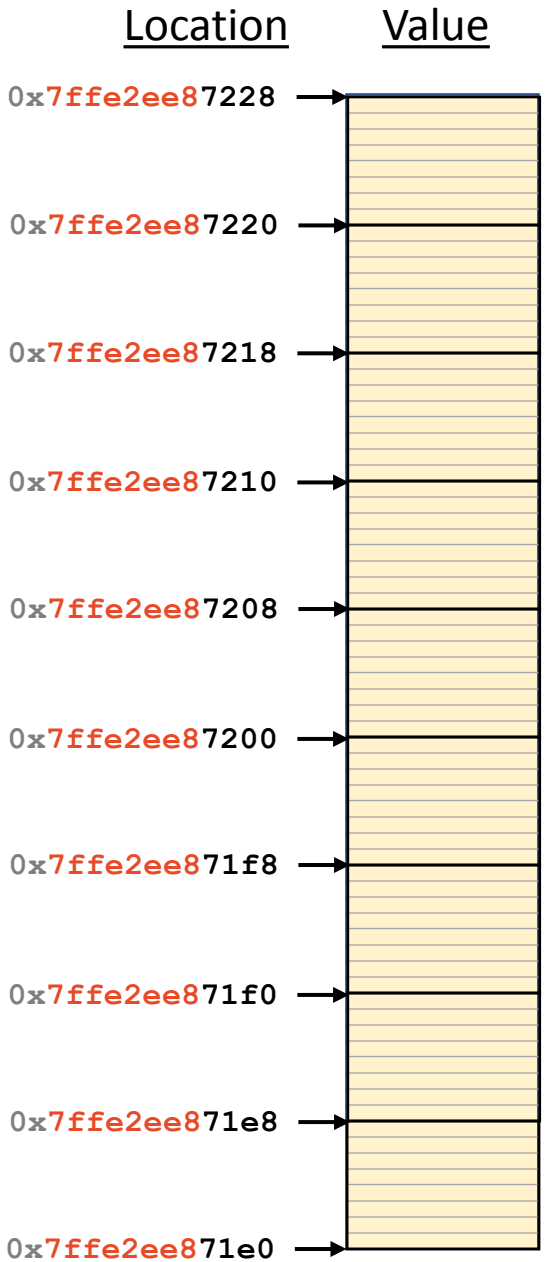
```
1  int main() {
2    int a;
3    int b = -3;
4    int c = 12345;
5
6    int *p = &b;
7
8    return 0;
9  }
```

sizeof-int.cpp

```
1  #include <iostream>
2
3  int main() {
4    std::cout << sizeof(int) << std::endl;
5    return 0;
6  }
```

sizeof-intptr.cpp

```
1  #include <iostream>
2
3  int main() {
4    std::cout << sizeof(int *) << std::endl;
5    return 0;
6  }
```

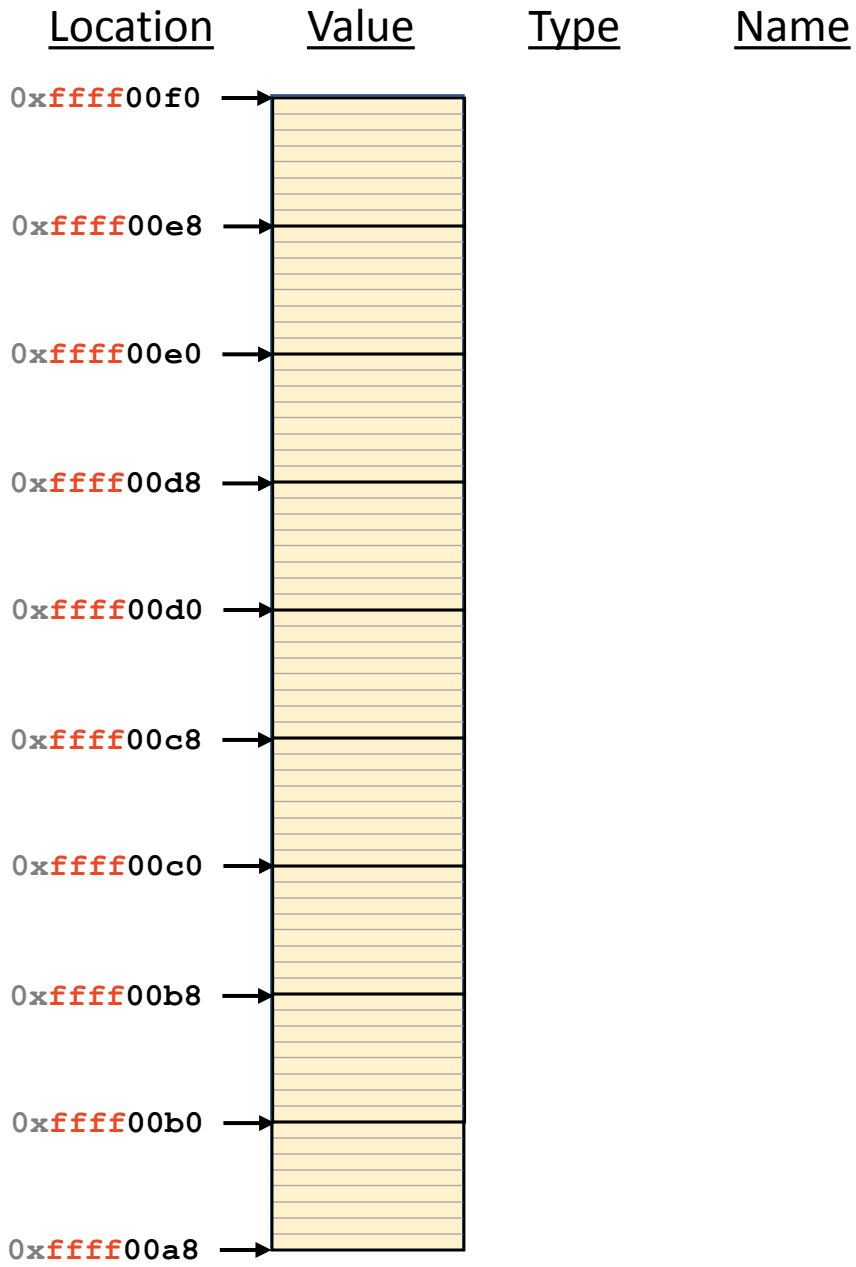


example1.cpp

```
1  int main() {
2    int a;
3    int b = -3;
4    int c = 12345;
5
6    int *p = &b;
7
8    return 0;
9  }
```

Real results when running on **linus.ews.illinois.edu**

```
&a: 0x7ffe2ee87218
&b: 0x7ffe2ee87214
&c: 0x7ffe2ee87210
&p: 0x7ffe2ee87208
```



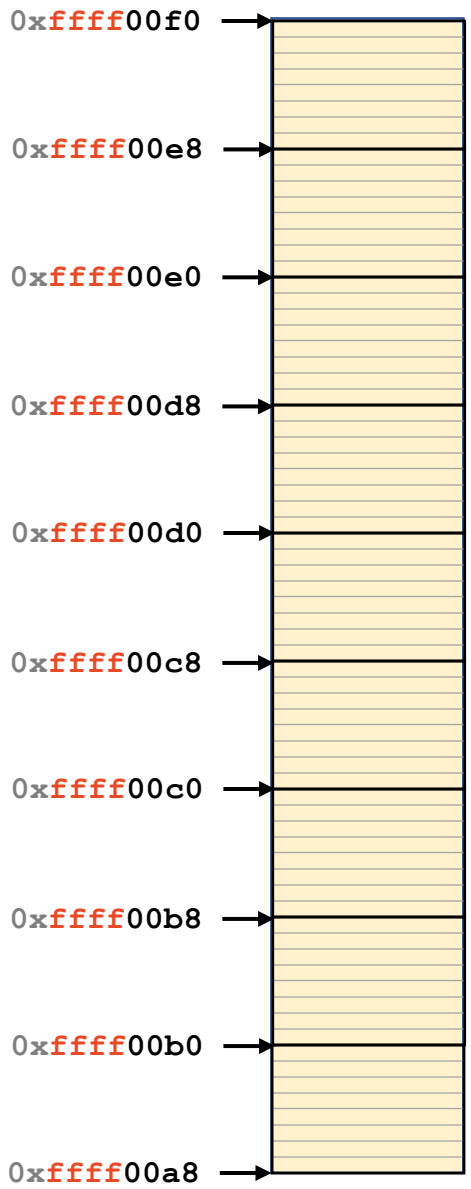
example2.cpp

```
1  #include "sphere.h"
2
3  int main() {
4    cs225::Sphere s;
5    cs225::Sphere *p = &s;
6
7    return 0;
8  }
9
```

sizeof-sphere.cpp

```
1  #include <iostream>
2  #include "sphere.h"
3
4  int main() {
5    std::cout << sizeof(cs225::Sphere) << std::endl;
6    std::cout << sizeof(cs225::Sphere *) << std::endl;
7    return 0;
8  }
```

Stack Frames



stackframe.cpp

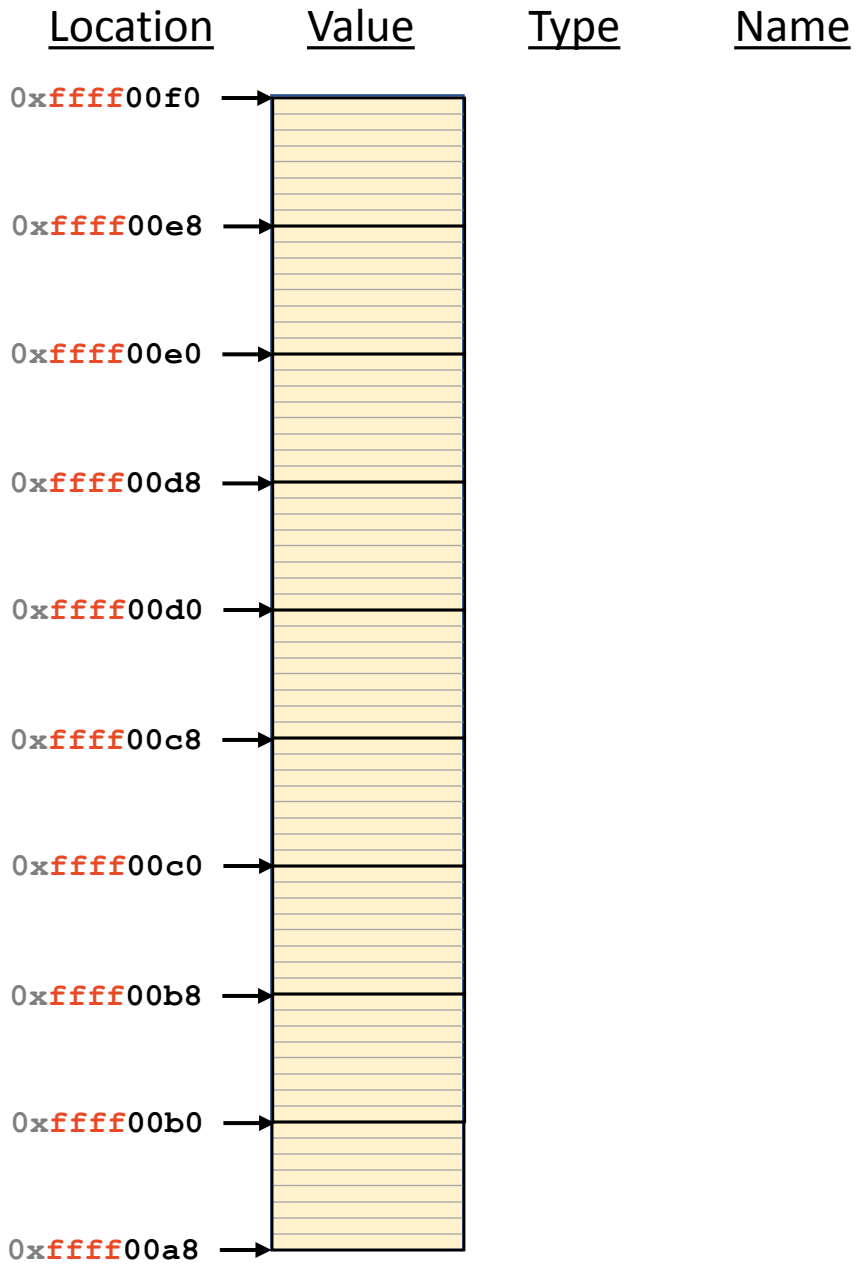
```
 1 int hello() {
 2   int a = 100;
 3   return a;
 4 }
 5
 6 int main() {
 7   int a;
 8   int b = -3;
 9   int c = hello();
10   int d = 42;
11
12   return 0;
13 }
```

Problems of the Day (POTD)

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

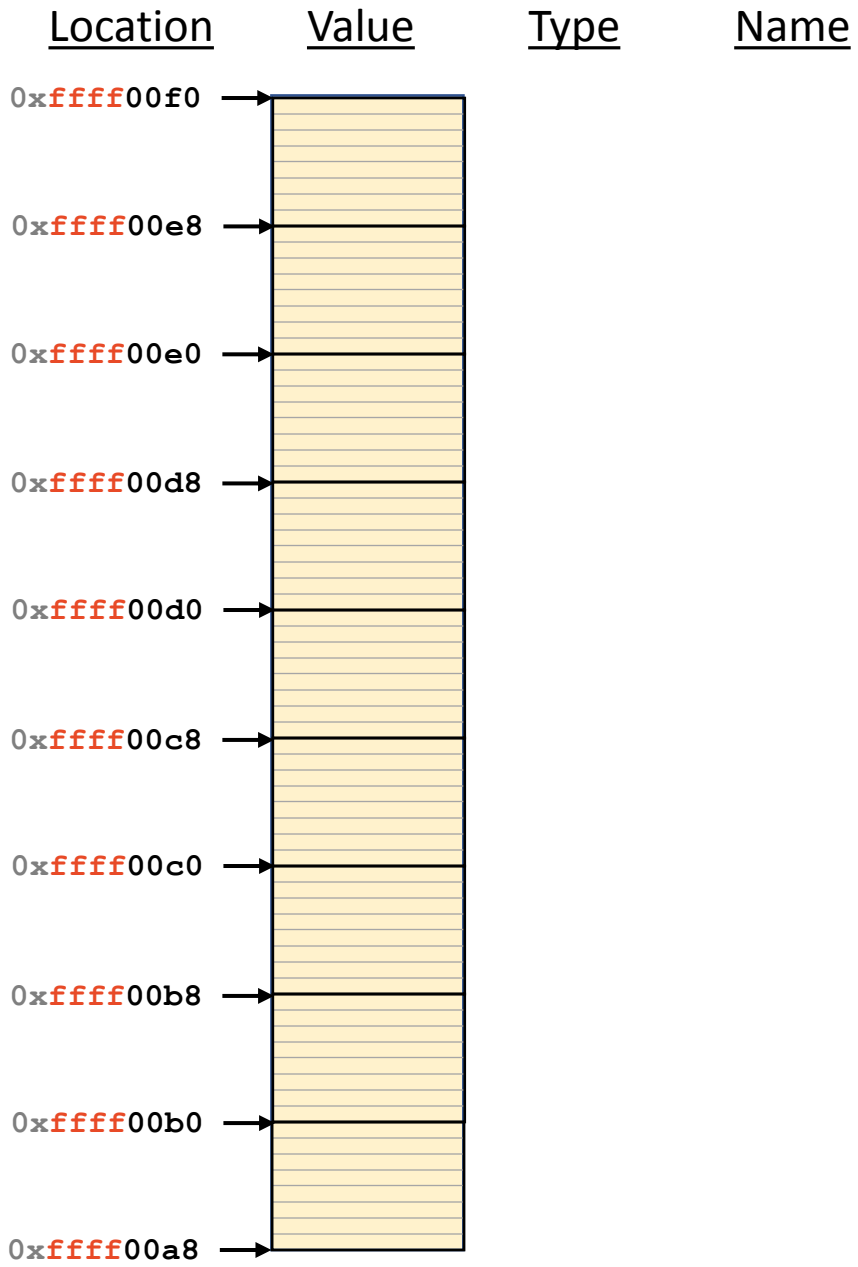
Each POTD is worth **+1** extra credit point, capped at **+40**.
(Course-wide, all extra credit is capped at +100.)

POTD#1 is available now, until 8:00am tomorrow morning when POTD#2 becomes available!



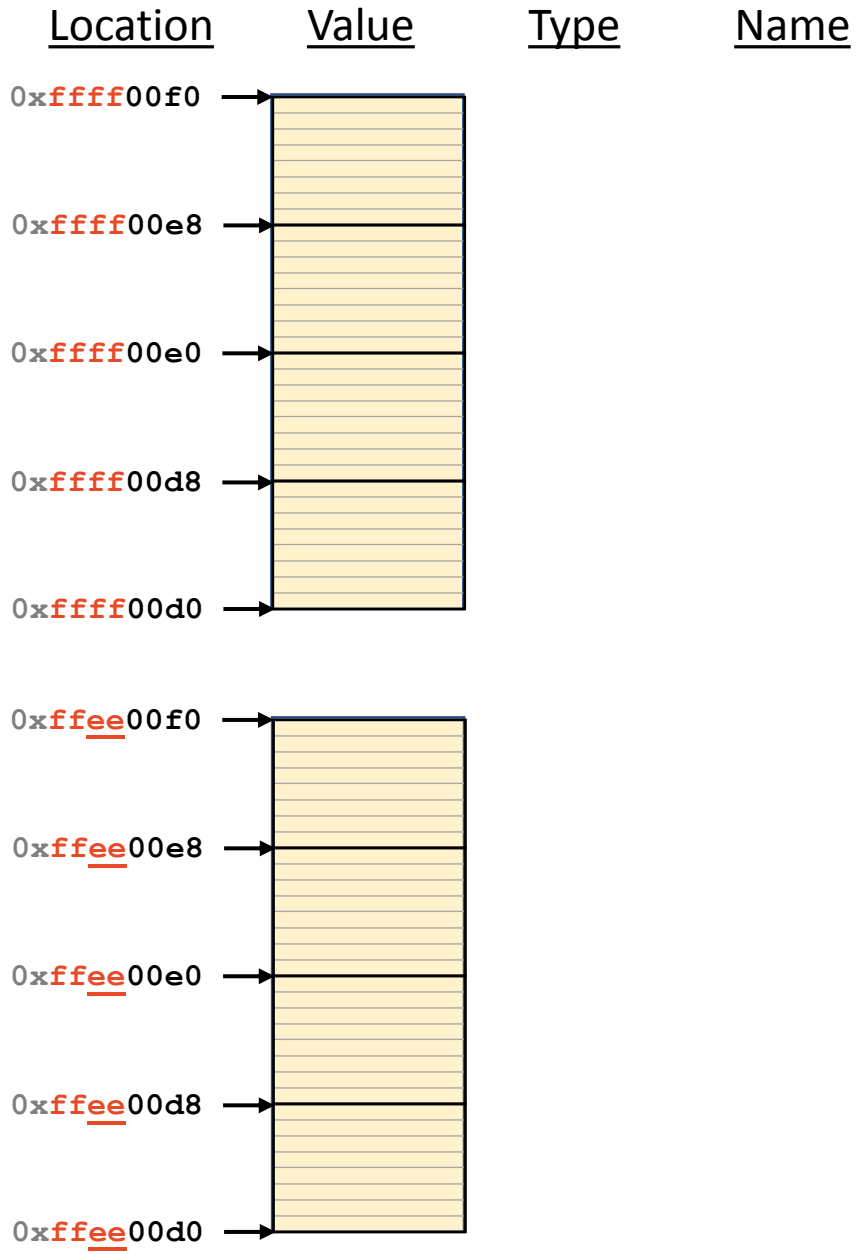
puzzle.cpp

```
 1 #include "sphere.h"
 2 using namespace cs225;
 3
 4 Sphere *CreateUnitSphere() {
 5   Sphere s(1);
 6   return &s;
 7 }
 8
 9 int main() {
10   Sphere *s = CreateUnitSphere();
11   double r = s->getRadius();
12   double v = s->getVolume();
13   return 0;
14 }
```



puzzle.cpp

```
 1 #include "sphere.h"
 2 using namespace cs225;
 3
 4 Sphere *CreateUnitSphere() {
 5   Sphere s(1);
 6   return &s;
 7 }
 8
 9 int main() {
10   Sphere *s = CreateUnitSphere();
11   double r = s->getRadius();
12   double v = s->getVolume();
13   return 0;
14 }
```



puzzle.cpp

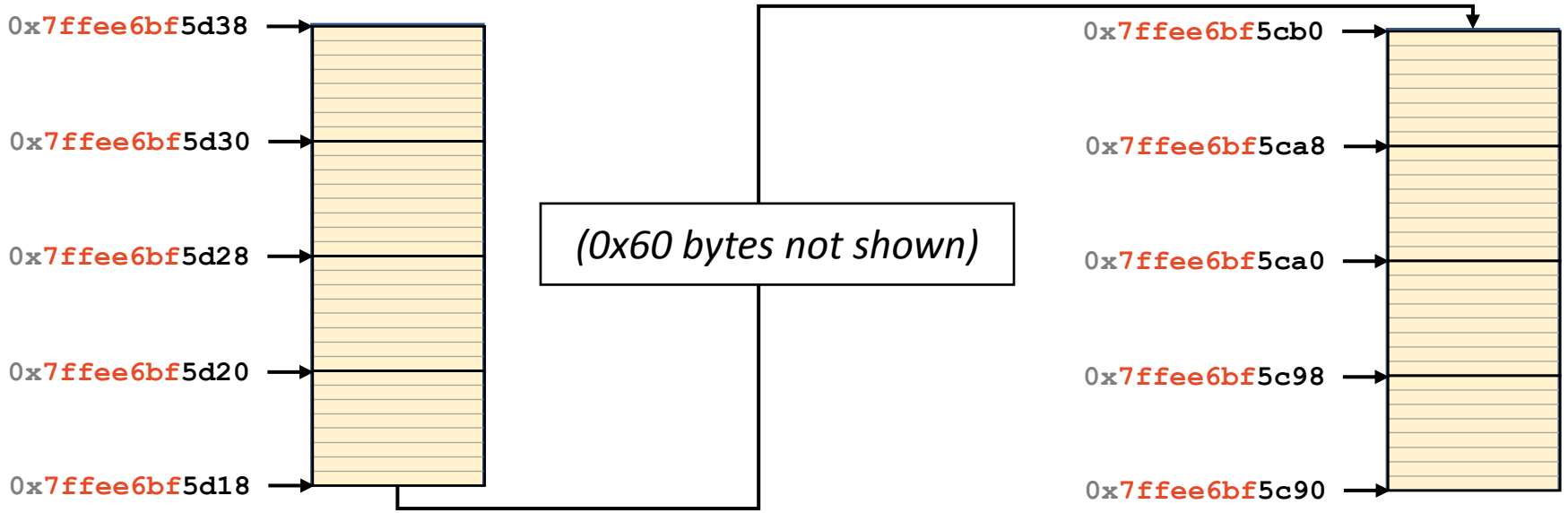
```
 1 #include "sphere.h"
 2 using namespace cs225;
 3
 4 Sphere *CreateUnitSphere() {
 5   Sphere s(1);
 6   return &s;
 7 }
 8
 9 int main() {
10   Sphere *s = CreateUnitSphere();
11   double r = s->getRadius();
12   double v = s->getVolume();
13   return 0;
14 }
```

What happens on a real system?

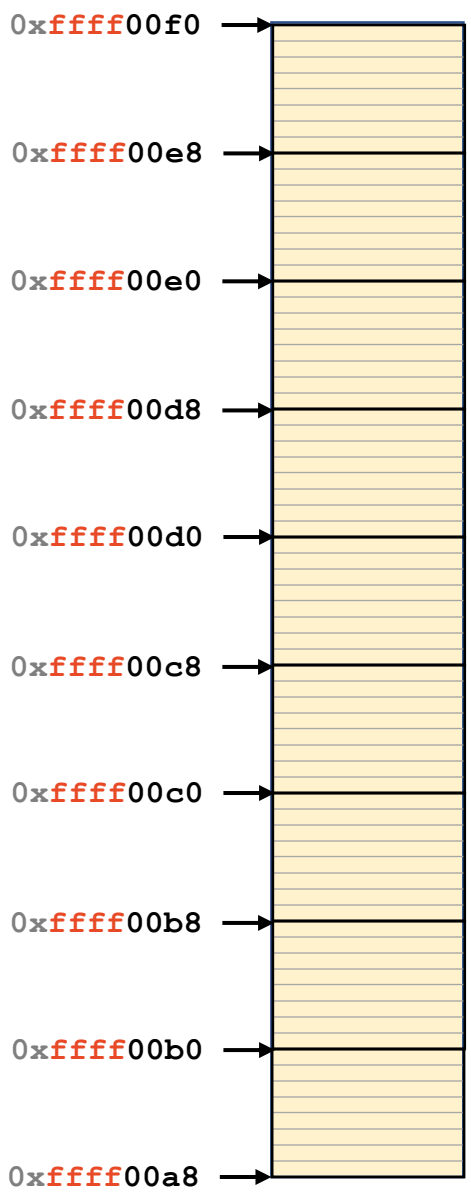
```
13  int main() {
14    Sphere *s = CreateUnitSphere();
15    cout << s->getRadius() << endl;
16    cout << "s->getRadius(): "
           << s->getRadius() << endl;
17    cout << "&s (main): " << &s << endl;
18    cout << " s (main): " <<  s << endl;
19    double r = s->getRadius();
20    cout << "&r (main): " << &r << endl;
21    cout << " r (main): " <<  r << endl;
22    double v = s->getVolume();
23    cout << "&v (main): " << &v << endl;
24    cout << " v (main): " <<  v << endl;
25  }
```

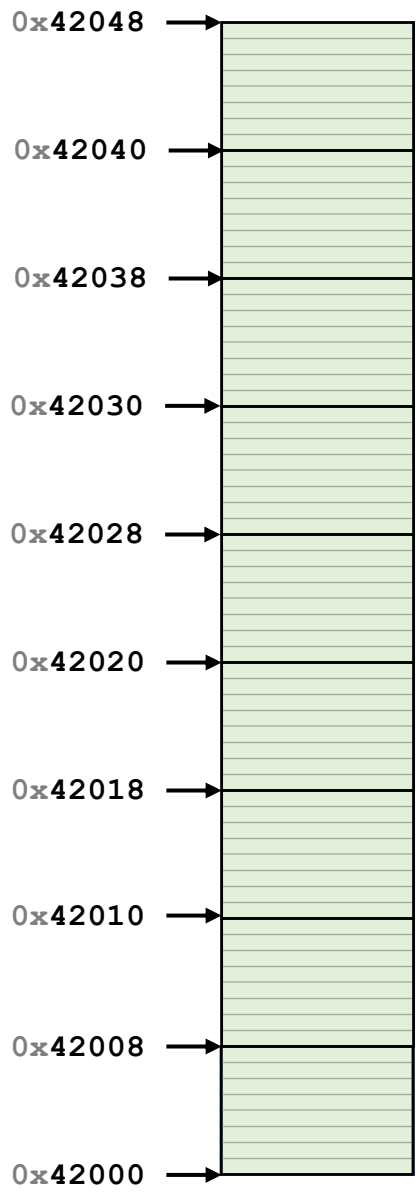
Real results when running on **linus.ews.illinois.edu**

```
&s (CreateUnitSphere): 0x7ffee6bf5ca8
1
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
```

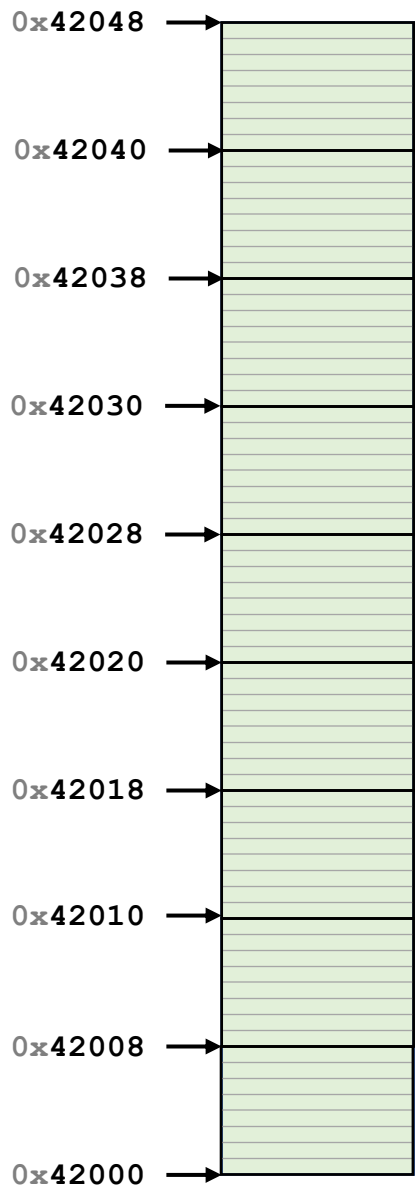


Stack Memory vs. Heap Memory





Heap Memory


0x42048
0x42040
0x42038
0x42030
0x42028
0x42020
0x42018
0x42010
0x42008
0x42000

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

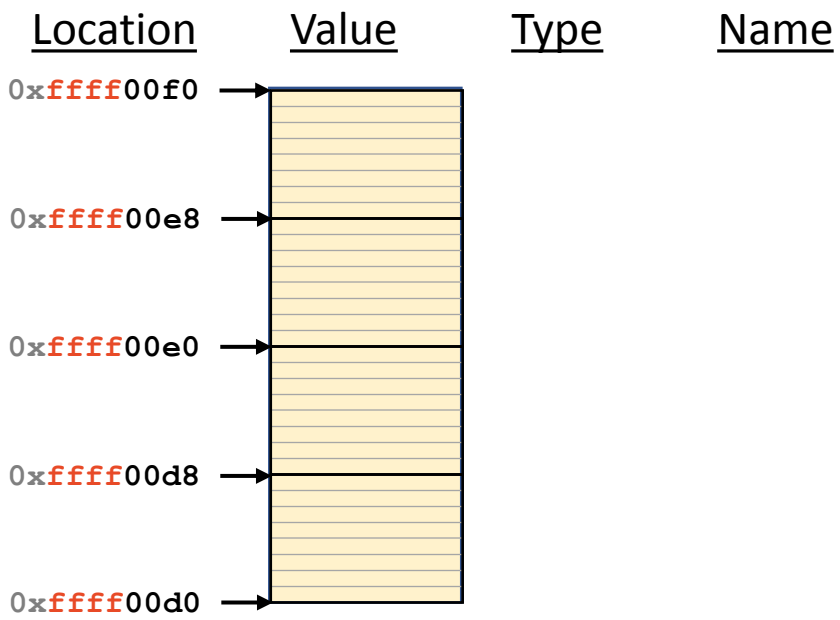
2. The only way to free heap memory is with the use of the **delete** keyword. Using **delete** will:

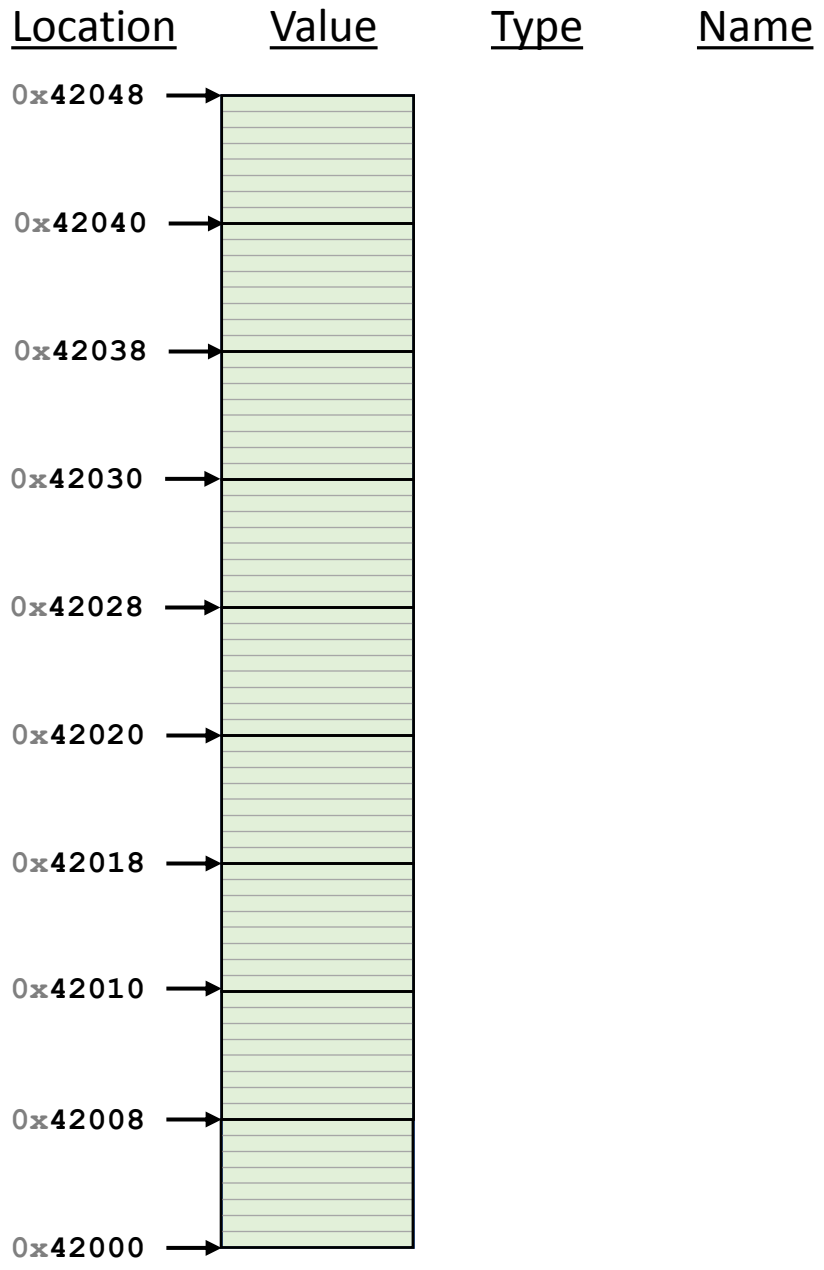
-
-

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”.

heap1.cpp

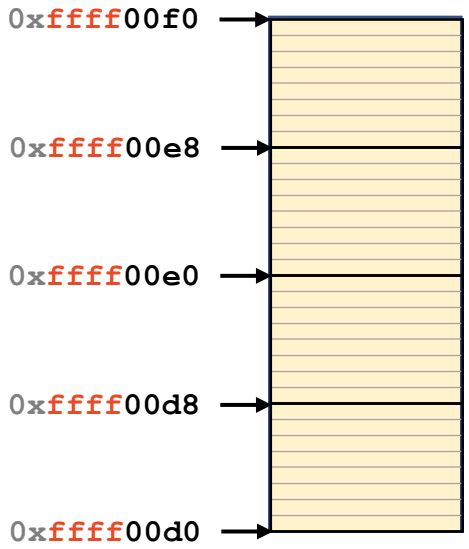
```
1  #include "sphere.h"
2  using namespace cs225;
3
4  int main() {
5    int *p = new int;
6    Sphere *s = new Sphere(10);
7
8    return 0;
9  }
```

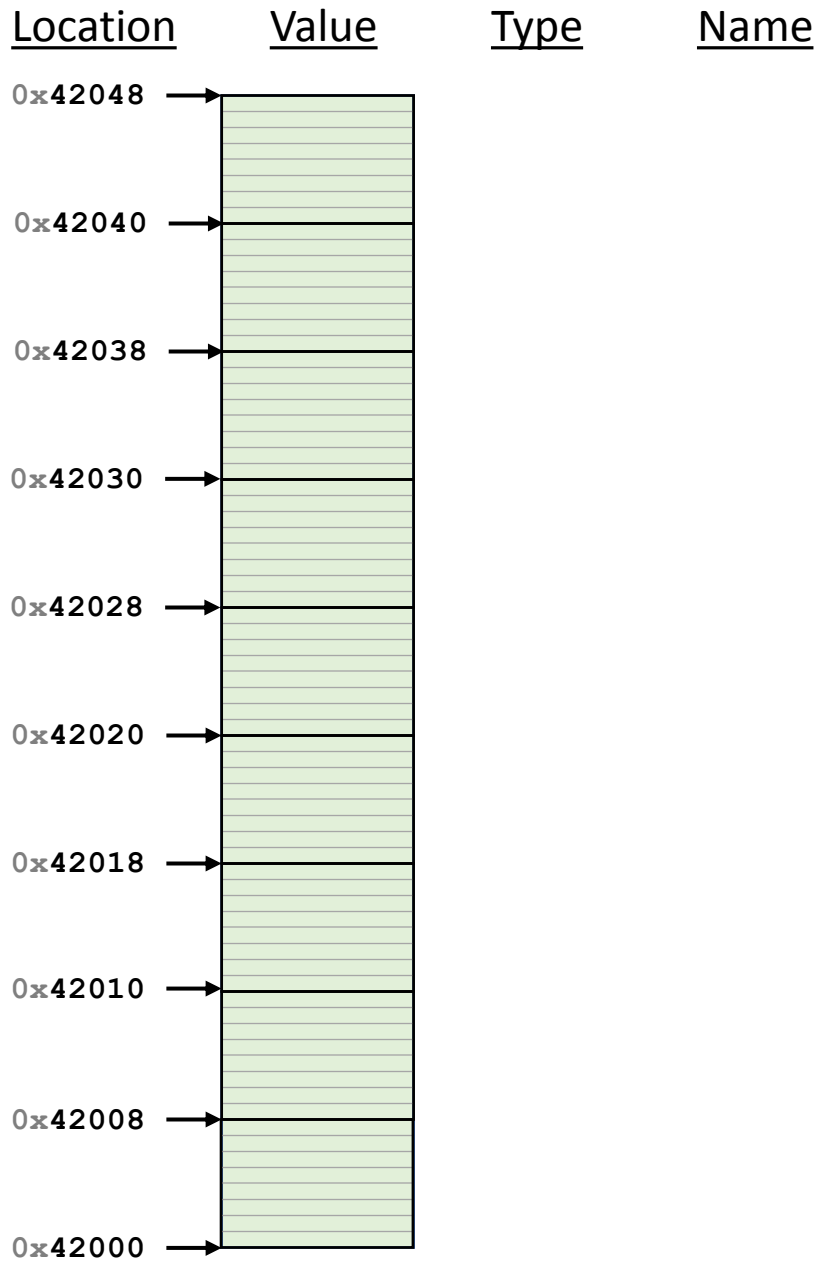




heap2.cpp

```
 1 #include "sphere.h"
 2 using namespace cs225;
 3
 4 int main() {
 5   Sphere *s1 = new Sphere();
 6   Sphere *s2 = s1;
 7
 8   s2->setRadius( 10 );
 9
10   return 0;
11 }
```





extra-puzzle1.cpp

```
 1 #include <iostream>
 2 using namespace std;
 3
 4 int main() {
 5   int *p;
 6   int x;
 7
 8   p = &x;
 9   x = 6;
10
11   cout << x << endl;
12   cout << p << endl;
13
14   return 0;
15 }
```

extra-puzzle2.cpp

```
 1 #include <iostream>
 2 using namespace std;
 3
 4 int main() {
 5   int *p, *q;
 6   p = new int;
 7   q = p;
 8   *q = 8;
 9   cout << *p << endl;
10
11   q = new int;
12   *q = 9;
13   cout << *p << endl;
14   cout << *q << endl;
15
16   return 0;
17 }
```

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)*

POTDs 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