Stuff++

Stack Variables

```c
#include <stdio.h>

int DoNothing(int a) {
  int b;
  b = a;
  return a;
}
```

Heap Variables

```c
#include <stdio.h>
#include <stdlib.h>

int DoNothing(int a) {
  int b;
  b = a;
  return a;
}

int DoNothing2(int a) {
  int* b;
  b = (int *)malloc(sizeof(int));
  *b = a;
  free(b);
  return a;
}
```

And now in C++

```
#include <iostream>

int DoNothing3(int a) {
  int* b;
  b = new int;
  *b = a;
  delete b;
  return a;
}
```

Arrays

```
int DoNothing4(int a) {
  const unsigned int ARR_SIZE = 3;
  int* b;

  // Allocate a new array
  b = new int[ARR_SIZE];

  // Fill array
  for (int i = 0 ; i < ARR_SIZE ; ++i) {
    b[i] = a;
  }

  // Print array
  for (int i = 0 ; i < ARR_SIZE ; ++i) {
    std::cout << b[i] << " ";
  }
  std::cout << std::endl;

  // Cleanup and return
  delete[] b;
  return a;
}
```

malloc vs. new

malloc	new
allocates memory	allocates memory
requires size and cast	does not require size and cast
provides uninitialized memory	calls the constructors
use in C code	use in C++ code

ctors and dtors

```
int DoNothing(int a) {
  int b;
  b = a;
  return a;
}

MyClass DoNothing(MyClass a) {
  MyClass b;
  b = a;
  return a;
}
```

ctors and dtors

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

class CtorDtorSpy {
public:
  CtorDtorSpy() {cout << "Ctor called. this = " << this << endl;}
  ~CtorDtorSpy() {cout << "Dtor called. this = " << this << endl;}
};

void DoNothing(void) {
  CtorDtorSpy b;
}

int main(void) {
  CtorDtorSpy a;
  DoNothing();
}
```

```
Ctor called. this = 0x7fff56249a38
Ctor called. this = 0x7fff56249a08
Dtor called. this = 0x7fff56249a08
Dtor called. this = 0x7fff56249a38
```

ctors and dtors

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

class CtorDtorSpy {
public:
  CtorDtorSpy() {cout << "Ctor called. this = " << this << endl;}
  ~CtorDtorSpy() {cout << "Dtor called. this = " << this << endl;}
};

int main(void) {
  CtorDtorSpy* a = new CtorDtorSpy[3];
  delete[] a;
}
```

```
Ctor called. this = 0x7fbdeb403978
Ctor called. this = 0x7fbdeb403979
Ctor called. this = 0x7fbdeb40397a
Dtor called. this = 0x7fbdeb40397a
Dtor called. this = 0x7fbdeb403979
Dtor called. this = 0x7fbdeb403978
```

ctors and dtors

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

class CtorDtorSpy {
public:
  CtorDtorSpy() {cout << "Ctor called. this = " << this << endl;}
  CtorDtorSpy(int i) {cout << "Ctor called with "<< i << ". this = " << this << endl;}
  ~CtorDtorSpy() {cout << "Dtor called. this = " << this << endl;}
};

int main(void) {
  CtorDtorSpy a(7);
}
```

```
Ctor called with 7. this = 0x7fff5309ba38
Dtor called. this = 0x7fff5309ba38
```

ctors and dtors and assignments

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

class CtorDtorSpy {
public:
  CtorDtorSpy() {cout << "Ctor called" << endl;}
  CtorDtorSpy(const CtorDtorSpy& other) {cout << "Copy ctor called" << endl;}
  ~CtorDtorSpy() {cout << "Dtor called" << endl;}
  CtorDtorSpy& operator=(const CtorDtorSpy& other) {cout << "Assignment called" << endl; return *this;}
};


int main(void) {
  CtorDtorSpy a;
  CtorDtorSpy b = CtorDtorSpy();
  CtorDtorSpy c(a);
  c = b;
}
```

```
Ctor called
Ctor called
Copy ctor called
Assignment called
Dtor called
Dtor called
Dtor called
```

* and &

declare a pointer	`int* p;`
dereference a pointer	`*p = 5;`
get memory address	`int i = 9; cout << &i;`
declare a reference	`int& r = i;`

const

```
const int * p;          // pointer can change, object is const
int const * p;          // less common, same as previous line

int * const p;          // pointer is constant, object can change
const int * const p; // both pointer and object are constant
```

```
class SomeClass {
  void DoesNotChangeX() const;
  int member_x;
}
```

Find the bug

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

const int* GetNumber(void) {
  const int THE_NUMBER = 5;
  return &THE_NUMBER;
}

int main(void) {
  const int* p = GetNumber();
  cout << *p << endl;
}
```

```
cpp_precept.cpp:8:11: warning: address of stack memory associated with local variable
'THE_NUMBER' returned [-Wreturn-stack-address]
```

Find the bug

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

int& GetNumber(void) {
  int THE_NUMBER = 5;
  return THE_NUMBER;
}

int main(void) {
  int i = GetNumber();
  cout << i << endl;
}
```

```
cpp_precept.cpp:8:11: warning: address of stack memory associated with local variable
'THE_NUMBER' returned [-Wreturn-stack-address]
```