Memory and Pointers

Stack/Heap/Global

- Stack
  - Allocated by context entry
  - Lifetime same as the function
- Heap
  - Allocated by explicit code (details later today)
  - Lifetime managed by explicit code (details later today)
- Global
  - Allocated by runtime
  - Lifetime the whole runtime of program

So Far

```
int x;
double probability;
bool feature_vector[28][28];
vector<ImageData> training_images;
ImageData tmp_image;
cin >> tmp_image;
training_images.push_back();
```

Pointers
(Dereference Operator *, Address of Operator &)

Pointers are just variables
- Store addresses

Declare in C++ as follows
```
int *ptr_x;
```

How do I set a pointer
```
ptr_x = ptr_y;
ptr_x = &x;
```

How do I access what a pointer points to?
```
*ptr_x = 42;
cout << *ptr_x;
```

MAN, I SUCK AT THIS GAME. CAN YOU GIVE ME A FEW POINTERS?
0x3A28213A 0x6339392C, 0x7363682E.
I HATE YOU.

What is the behavior?

```
int *ptr;
int val;
ptr = &val;
*ptr = 10;
cout << val;
```

What probably happens?

A. 10 printed
B. Some address printed
C. Some unknown value printed
D. Segfault and crash

Explicit Dynamic Allocation

- new
  - allocates memory and constructs objects returning the address
  - int *heap_int = new int;
  - Can allocator arrays
  - int *heap_array = new int[10]
- delete
  - Releases memory allocated with new
  - `delete heap_int;`
  - Must specify when releasing arrays
  - `delete[] heap_array;`

What is the behavior?

```
int *heap_x;
int *heap_y;
heap_x = new int;
heap_y = heap_x;
*heap_y = 10;
cout << *heap_x;
```

What probably happens?

A. 10 printed
B. Some address printed
C. Some unknown value printed
D. Segfault and crash

What is the behavior?

```
int *heap_x;
int *heap_y;
heap_y = heap_x;
heap_y = new int;
*heap_y = 10;
cout << *heap_x;
```

What probably happens?

A. 10 printed
B. Some address printed
C. Some unknown value printed
D. Segfault and crash

Passing Arguments

- By value
  - Make a copy
- By reference
  - Like Java objects
- By pointer
  - Pass a copy of the pointer

What happens?

```
void fn(int x) {
     x = 10;
}

int main() {
     int x = 200;
     fn(x);
     cout <<  x;
}
```

What probably happens?

A. 10 printed
B. 200 printed
C. Won't compile
D. Some unknown value printed
E. Segfault and crash

What happens?

```
void fn(int &x) {
     x = 10;
}

int main() {
     int x = 200;
     fn(x);
     cout <<  x;
}
```

What probably happens?

A. 10 printed
B. 200 printed
C. Won't compile
D. Some unknown value printed
E. Segfault and crash

What happens?

```
void fn(int *x) {
     x = 10;
}

int main() {
     int x = 200;
     fn(x);
     cout <<  x;
}
```

What probably happens?

A. 10 printed
B. 200 printed
C. Won't compile
D. Some unknown value printed
E. Segfault and crash

What happens?

```
void fn(int *x) {
     x = 10;
}

int main() {
     int x = 200;
     fn(&x);
     cout <<  x;
}
```

What probably happens?

A. 10 printed
B. 200 printed
C. Won't compile
D. Some unknown value printed
E. Segfault and crash