CS 161
Intro to CS I
Pointers and Static vs. Dynamic Memory
Variables vs. Pointers

• Value Semantics
  – Values stored directly
  – Copy of value is passed
    int i, j=2;
    i=j;

• Pointer Semantics
  – Address to variable is stored
  – Copy of address is passed
    int *i, j=2;
    i=&j;
What if we don’t have the j?

• We need to create the address space.
• How do we do this?
  – **new** type;
• For example:
  int *i = NULL;
i = new int;  //new returns an address
  *i = 10;
• [http://cslibrary.stanford.edu/104/](http://cslibrary.stanford.edu/104/)
Static vs. Dynamic

- **Static Semantics**
  - Assign address of variable
    ```
    int *i, j=2;
    i=&j;
    ```

- **Dynamic Semantics**
  - Create memory
  - Assign memory to pointer
    ```
    int *i=NULL;
    i=new int;
    *i=2;
    ```
Understanding Pointers

- Pointers to the Stack

- Pointers to the Heap
References vs. Pointers Demo...
Creating Memory in Functions

Advantages to Dynamic Memory
int *i=NULL;//created in main function

i = create_mem();//call in main
int * create_mem() {
    return new int;
}

OR
create_mem(&i);//call in main void
create_mem(int **m) {
    *m = new int;
}
Demo...
What About Memory Leaks?

- What happens here...

...  
int main () {  
    int *i=NULL;  //created in main function  
    while(1) {  
        i = create_mem();  //call in main  
    }  
}  

int * create_mem() {  
    return new int;  
}
Fixing Memory Leaks...

• What happens here...

...  
int main () {
    int *i=NULL;  //created in main function
    while(1) {
        i = create_mem();  //call in main
        delete i;  //free memory that i points to, preventing mem leaks
    }
}

int* create_mem(){
    return new int;
}