CS 161
Intro to CS I

More About Functions:
Default Values, Overloading, and
References vs. Pointers
Odds and Ends...

• Assignment #4 posted
• Demo Assignment #3
• Pythontutor.com (good way to visualize code)
```cpp
#include <iostream>

using std::cout;
using std::endl;

int pwr(int, int n=1); // Example of default args

int main() {
    int base=2, expn=8;

    cout << "The power function: " << pwr(base, expn) << endl;
    cout << "The power function: " << pwr(base) << endl;

    return 0;
}

int pwr(int x, int n) {
    int num=1;
    for(int i=0; i < n; i++) {
        num*=x;
    }
    return num;
}
```

C++ Function Overloading

• Multiple functions w/ same name
• Arguments determine function
• Default Args can be done w/ overloading
• Example: pow()
void swap(int, int);
int main() {
    int a=5, b=10;
    swap(a, b);
    cout << "a: " << a << "b: " << b;
}
void swap(int x, int y) {
    int temp = x;
    x = y;
    y = temp;
}
C++ Pass by Reference

void swap(int &, int &);

int main() {
    int a=5, b=10;
    swap(a, b);
    cout << "a: " << a << " b: " << b;
}

void swap(int &x, int &y) {
    int temp = x;
    x = y;
    y = temp;
}
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;
    ```
C/C++ Pointers

void swap(int *, int *);
int main() {
    int a=5, b=10;
    swap(&a, &b);
    cout << "a: " << a << "b: " << b;
}
void swap(int *x, int *y) {
    int temp = *x;
    *x = *y;
    *y = temp;
}
```cpp
#include <iostream>

using namespace std;

//overloaded functions with different parameter types
void swap(int &, int &);
void swap(int *, int *);

int main() {
    int x = 3, y = 2;
    cout << &x << endl;  //gives us address of where main's x is memory
    cout << &y << endl;  //gives us address of where main's y is memory
    swap(x, y);          //call swap with reference paramters
    cout << x << "  " << y << endl;
    swap(&x, &y);        //call swap with pointer parameters, explicitly pass address
    cout << x << "  " << y << endl;
    return 0;
}
```
```cpp
4  // overloaded function with other swap that has reference types, rather than
5  // pointer types. Pass by reference example where we can change main's
6  // x and y value in swap. x and y used in this function always refer to x
7  // and y in main/arguments that they were made references to.
8  void swap(int &x, int &y) {
9      int temp=x;
10     cout << &x << endl;  // should get x in main address
11     cout << &y << endl;  // should get y in main address
12     x=y;
13     y=temp;
14 }
15  // overloaded function with other swap that has pointer types, rather than
16  // reference types. Pass by pointers example where we can change main's
17  // x and y value in swap. x and y used in this function always refer to
18  // local x and y, unless explicitly dereferenced to take you to main's
19  // x and y.
20  void swap(int *x, int *y) {
21      int temp=*x;
22     cout << &x << endl;  // my own local x address
23     cout << &y << endl;  // my own local y address
24     cout << x << endl;  // contents of local x is address of x in main
25     cout << y << endl;  // contents of local y is address of y in main
26     *x=*y;  // explicitly dereference to take you to the address
27     *y=temp;
28 }
Pointer and References Cheat Sheet

• *
  – If used in a declaration (which includes function parameters), it creates the pointer.
    • Ex. int *p; //p will hold an address to where an int is stored
  – If used outside a declaration, it dereferences the pointer
    • Ex. *p = 3; //goes to the address stored in p and stores a value
    • Ex. cout << *p; //goes to the address stored in p and fetches the value

• &
  – If used in a declaration (which includes function parameters), it creates and initializes the reference.
    • Ex. void fun(int &p); //p will refer to an argument that is an int by implicitly using *p (dereference) for p
    • Ex. int &p=a; //p will refer to an int, a, by implicitly using *p for p
  – If used outside a declaration, it means “address of”
    • Ex. p=&a; //fetches the address of a (only used as rvalue!!!) and store the address in p.