CS 162
Intro to CS II

Exceptions and Templates
Error Handling

• Prevent it from happening
  – Checks before using it
• Let it happen and catch it
  – Exceptions
Exception Handling

- try { ... } block
- catch(exception &e) { ... } block
- Existing Exceptions:
  - Out of Range...
  - Bad memory allocation...
- Order matters!!!
Throwing Your Own Exception

try {
    throw variable/object;
}

catch(var_obj_type e) {
    ...
}
Programming Demo

```cpp
#include <stdexcept>
// #include <exception> // Don't need this when you include stdexcept
#include <iostream>
#include <string>
using namespace std;

int main(){
  string s;
  int n;

  try {
    throw n;
    cout << s.at(0) << endl;
    cout << s[0] << endl;
  }
  catch(int &e) { cout << s[0] << "int thrown!" << endl; }
  catch(exception &e) {
    cout << "General issue!" << endl;
  }
  catch(out_of_range &e) {
    cout << "You accessed something out of range!" << endl;
  }
  cout << "here";
```
Function Throw Exceptions

```c
return_type func_name(...) throw (type1, type2, ...);
int main() {
    try { func_name(); } 
    catch(type1 e) { ... } 
}
return_type func_name(...) {
    ... 
    throw type1; 
    ... 
}
```

Why Function Templates?

//at least C++ has overload
void swap(int &, int &);
void swap(char & char &);
...
void swap(int &a, int &b){
  int temp = a;
  a=b;
  b=temp;
}
void swap(char &a, char &b) {
  char temp = a;
  a=b;
  b=temp;
}
// Have to have this header

template<typename T>
void swap(T &, T &);
...

template<typename T>
void swap(T &a, T &b){
    T temp = a;
    a=b;
    a=b;
    b=temp;
}
When can you get into trouble?

// Have to have this header

template<typename T>
void func(T, T, int);

...

template<typename T>
void func(T a[], T b[], int size){
  // a is already a reference
  // what if we wanted to swap values in arrays
}

OSU Oregon State University
Why make a class templates?

• What example can we use? Vectors!!!!
Using a vector

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

int main()
{
    string s;
    int n;
    vector<int> v(2); // make a vector to hold 2 ints
    v[0]=1;
    v[1]=2;
    cout << v.size() << endl;
    v.push_back(3); // push the number 3 to the back of list
    cout << v.size() << endl;

    try {
        throw n;
        cout << s.at(0) << endl;
        cout << s[0] << endl;
    } catch ......
```

Class Templates

// Have to have this header

template<typename T>
class vector {
    public:
        vector();
        ~vector();
        void push_back(T);
    private:
        T *v;
};
#include "./vector.h"

vector::vector(){
    v=NULL;
}

vector::~vector(){
    delete [] v;
}

template<typename T>
void push_back(T element){
    ...
}