Lab 12

Part 3:

i. *Which exception handler is called?*
The following exception handler is called:

```
catch (...)

{

    cerr << "exception\n";

    return 2;
}
```

ii. *Based on this observation, what happens if the exception handler nearest to the try block does not contain an exception handler for the thrown object?*
If an exception is thrown, every catch block is checked for the exception.

iii. *What does the exception handler with the ellipsis (...) catch?*
All errors besides runtime error and someother_error.

Part 4:

i. *Which exception handler is called?*
The val_not_found exception is called.

ii. *Based on this observation, what happens if there is an exception handler nearest to the try block for the thrown object?*
The program will check that exception handler and attempt to catch the thrown object.

<u>Changed code:</u>

```
#include <iostream>

#include <vector>

#include <stdexcept>

// creates two new types... don't worry about this syntax, but understand that we can make objects
of type someother_error and of type val_not_found... that we can throw... and subsequently catch.

class someother_error {};

class val_not_found {};
```

```
using namespace std;


int& find_int(vector<int>&, int);


int main()

try

{

    vector<int> vint {2, 4, 6, 8};

    cout << "Enter value to find: ";

    int val;

    cin >> val;

    try {

        int& i = find_int(vint, val);

    } catch(val_not_found &e) {

        cerr << "val_not_found" << endl;

        return 7;

    } catch (someother_error &e) {

        cerr << "exeption: someother_error" << '\n';

}

    return 0;

} catch (runtime_error& e)

{

    cerr << e.what() << '\n';

    return 1;
} catch (...)

{

   cerr << "exception\n";

   return 2;

}



int& find_int(vector<int>& vint, int val)

{

   for (decltype(vint.size()) i = 0; i < vint.size(); ++i)

      // decltype as used in this expression essencially says that I want the base type of i to be
whatever type the value returned by vint.size() is.

      if (val == vint.at(i))

         return vint.at(i);

   throw val_not_found{};

}
```