Hieu Tran

5/14/15

**Homework 12**

**2.**

**#include** <iostream>

**#include** <memory>

**#include** <string>

**#include** <stdexcept>

**using** **namespace** std;

**int** **main**(**int** argc, **char**\* argv[]) {

**try** {

cout << "Enter a number less than 10: " << **endl**;

unique\_ptr<**int**> x(**new** **int**());

cin >> \*x;

**if** ((\*x) >= 10) {

**throw** out\_of\_range("listen to instruction dummy!");

}

}

**catch** (out\_of\_range &ex) {

cout << "You didn't listen!" << **endl**;

}

}

**3.**

**#include** <vector>

**#include** <numeric>

**#include** <iostream>

**using** **namespace** std;

**int** **Add**(**int** a, **int** b) {**return** a + b;};

**int** **Min**(**int** a, **int** b) {**return** a <= b ? a : b;};

**bool** **Any**(**int** a, **int** b) {**return** a || b;};

**bool** **All**(**int** a, **int** b) {**return** a && b;};

**int** **main**(**int** argc, **char**\* argv[]) {

vector<**int**> values = {8, 6, 7, 5, 3, 0, 9};

vector<**bool**> flags = {**true**, **true**, **true**, **false**, **true**, **false**};

cout << accumulate(values.begin(), values.end(), 0, Add) << **endl**;

//EXAMPLE ANSWER: output = 38, which is the SUM of all numbers.

cout << accumulate(values.begin(), values.end(), 214748367, Min) << **endl**;

// Output: 0, which is the smallest number comparing to 214748367.

cout << accumulate(flags.begin(), flags.end(), **false**, Any) << **endl**;

// Output: 1 (true), which is a true statement of all bool statements.

cout << accumulate(flags.begin(), flags.begin() + 3, **true**, All);

// Output: 1 (true), which is a true statement up of the first 3 elements of

// bool statements.

}

**4.**

**#include** <numeric>

**#include** <math.h>

**#include** <functional>

**#include** <iostream>

**using** **namespace** std;

**int** **Pow**(**int** a, **int** b) {**return** **pow**(a, b);};

**int** **main**(**int** argc, **char**\* argv[]) {

function<**int**(**int**, **int**)> func = Pow;

cout << func(func(3,2), func(4,1)) << **endl**;

//Output: 6561

}

**5.**

**a.**integer

**b.** signed long integer

**c.** character pointer

**d.** Rational class pointer

**e.** integer

**f.** vector<int>

**6.**

a. True

b. False

c. False

d. False

**7.**

a. **double** g = 8.5;

**int** i = **static\_cast**<**int**> (g);

b. Animal \*a;

Cat \*c = dynamic\_cast<Cat \*> (a);

c. Rational r;

string s = static\_cast<string>(r);