- Project 2 Has Been Released On The CCLE. Please Go Into The Assignment And Scroll Down For Your Score And Feedback.

The Average Was 78 (Out Of 100) But Half The Scores Were 80 Or Higher. Nice Job!

Sorry, The my.ucla.edu Gradebook Is Not Working...

  Sample Problems Have Been Released In The CCLE For You To Ponder...
  Out Of 100 Points But Worth 25% Of Your Total Grade
  No Devices, No Phones, Closed Book
  Please Bring A Photo ID
  No Scantron Needed
- Expect To Write Small Amounts Of Code
- Covers:
  Variables and Datatypes
  Boolean Conditions
  < > <= >= == != && || !
  Flow Of Control Statements
  if else if-else if-else do-while for while switch
  Functions
pass-by-value
pass-by-reference
overloading
Not Very Much On: std::string
I Have Released Sample Review Problems We Will Discuss Wednesday.

- Discussion Board Questions
  What About Leading Zeros, As In: "t0000070c0000069+000001"
  You Are Free To Handle This However You Wish, As Either Legal or Illegal. The Scorer Will Not Send You Any Leading Zeros. Please Document Your Design Decision And Move On
  What About * and /, As In: "t70c69*12"  ILLEGAL! Only + And - Allowed, If Any
  What About Matching Values, As In: "t70c70"  "t70h70"  LEGAL.
  Cooling Set But System Is Not Running
  What About Zero Increments, As In: "t70c69+0+1+1"  ILLEGAL!
  Increment Not Allowed To Be Zero
  What About Negative Temperatures, As In "t-20h-100"  ILLEGAL!
  All Temperature Settings Will Be Zero Or More

Back To Functions...

Parameter Passing Schemes
CALLER
The pile of code that invokes the function
CALLEE
The pile of code that gets called

<table>
<thead>
<tr>
<th>Pass-By-Value</th>
<th>Declared As:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>void foo(int i);</td>
</tr>
<tr>
<td></td>
<td>void foo( int );</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Called As:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>foo( 12 );</td>
</tr>
<tr>
<td></td>
<td>foo( 'A' );</td>
</tr>
<tr>
<td></td>
<td>foo( j );</td>
</tr>
<tr>
<td></td>
<td>foo( j+1 );</td>
</tr>
</tbody>
</table>

"Very Safe" Programming
- Nothing The Callee Does Affects The Caller's Variable Value
- The Only Communication Between The Caller And The Callee Will Be With The Return Argument, If Not void And Captured By The Caller
- What Is Sent Is An "R-Value" Of The Desired Type
What Arrives Is An Independent Copy Of The Caller's Original Value - The Callee Can Change The Value But The Caller's Variable Will Not Change - INBOUND

<table>
<thead>
<tr>
<th>Pass-By-Reference</th>
<th>Declared As:</th>
<th>Called As:</th>
<th>Means:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>void foo( int &amp; a );</td>
<td>foo( i ); foo( j );</td>
<td>&quot;unsafe&quot; can cause some &quot;side effects&quot; &quot;strict&quot;</td>
</tr>
<tr>
<td></td>
<td>Double d;</td>
<td>foo( d );</td>
<td>The Caller Can Change The Value Of The Caller's Variable What Needs To Be Passed Is The Exact Right Matching Type l-value Of The Right Type Variable Of The Right Type No Copy The &quot;Actual Thing&quot; Is What Is Being Sent If You Want The Caller To See The Changed Value, This Is The Way You Send INBOUND - OUTBOUND</td>
</tr>
</tbody>
</table>

```cpp
#include <cassert>
// this is really going to work...
void swap( int& i, int& j );
// swap function is taking two ints by reference

int main()
{
    int a = 12, b = 13;
    swap( a, b );  /// a = 13, b = 12
    cout << "a=" << a << " b=" << b << endl;
    assert( a == 13 );  /// true statement
    assert( b == 12 );
```
return( 0 );

void swap( int& i, int& j )   /// make a copy!
{
    int temp = i;
    i = j;
    j = temp;
    // I = 13
    // j = 12
}

#include <cassert>
bool isValidThermostatString( string commands );

int main( )
{
    assert( isValidThermostatString( "" ) == false );
    assert( !isValidThermostatString( "     " ) );
    assert( !isValidThermostatString( "zzzzzz" ) );
    assert( !isValidThermostatString( "T" ) );
    assert( isValidThermostatString( "t70" ) );
    return( 0 );
}

bool isValidThermostatString( string commands )
{
    return( false );
}

void printArray( const int array[ ], int size )
{
    for (l = 0; l < size; l++)
        cout << "a[ I ] = " << a[ l ] << endl;
    a[ 0 ] = 4;    /// break....
}
// Project 2
// By Howard A. Stahl
#include <iostream>
#include <string>
#include <climits>
using namespace std;

void promptForInput(string& name, string& texting, string& web, string& wifi, int& minutes);
bool isValid(string name, string textingAnswer, string webAnswer, string wifiAnswer, int talkingAmt,
string& errorMessage);
bool wantsIt(string service);
double calculateCost(string texting, string web, string wifi, int minutes);
void showOutput(string name, double amount);
void showError(string errorMessage);

#include <cassert>
bool isValidThermostatString(string commands);
int number(string s, int& pos); // how many letters in the string it just ate....

int main()
{
    int i = 0;
    assert(number("", i) == -1);
    assert(number("123a456", i) == 123);
    // position 0123456
    assert(i == 3);
    i = 1;
    assert(number("123", i) == 23);
    assert(i == 3);
    i = 2;
    assert(number("123", i) == 3);
    i = 3;
    assert(number("123", i) == 3);
    i = 4;
    assert(number("123", i) == -1);
    i = 4;
    assert(number("123", i) == -1);
    i = 0;
assert(number("Howard", i) == -1);
assert(number("1900 Pico Boulevard", i) == 1900);

assert(isValidThermostatString("") == false);
assert(!isValidThermostatString("      ") == false);
assert(!isValidThermostatString("zzzzzz"));
assert(isValidThermostatString("T"));
assert(isValidThermostatString("t70"));
assert(isValidThermostatString("t777770");
assert(!isValidThermostatString("t-70");

return(0);
}

int number(string s, int& pos)
{
    int result = 0;
    // walk letter-by-letter, '0' - '9'
    // not worry about a + -
// starting from the beginning pos: 0
// int pos = 0;
// digit character
// problem: be sure we don't walk off the string
// problem: no digit character...
// test for a non-digit....
if (s.length() > pos)
{
    if (s[pos] < '0' || s[pos] > '9')
    {
        // failure
        result = -1;
    }
    else
    {
        while (s[pos] >= '0' && s[pos] <= '9')
        {
            // accumulate into result the ongoing value...
            // 1900           1     9    0
            result = result * 10 + (s[pos] - '0');
            pos = pos + 1;
            if (pos == s.length()) break;
        }
    }
}
else
{
    result = -1;
}

int mainYYY() {
    // driver code that uses the newly declared stubbed out functions
    string name("");
    string texting("");
    string web("");
    string wifi("");
    int minutes(0);
    double cost(0.0);
    string error("");

    promptForInput(name, texting, web, wifi, minutes);
    if (isValid(name, texting, web, wifi, minutes, error)) {
        cost = calculateCost(texting, web, wifi, minutes);
        showOutput(name, cost);
    } else {
        showError(error);
    }
    return(0);
}

int mainX() {
    /* variables for the data read from the user */
    string name("");
    bool texting(false); /* unlimited texting desired? */
    bool web(false); /* unlimited web desired? */
    bool wifi(false); /* wifi hotspots desired? */
    double talkingAmt; /* talking minutes this month */
    bool valid(true); /* is all the data valid?? */
    bool showedAnError(false); /* did I already print an error message?? */
    double cost(0.0);

    const string YES = "Yes";
    const string NO = "No";
    string textingAnswer("");
    string webAnswer("");
    string wifiAnswer("");

    // Prompt for the data
    cout << "Customer Name? ";
    getline(cin, name);
    cout << "Want unlimited texting? ";
    getline(cin, textingAnswer);
    cout << "Want unlimited web? ";
    getline(cin, webAnswer);
    cout << "Want wifi hotspots? ";
    getline(cin, wifiAnswer);
    cout << "How many minutes talking did you use this month? ";
    cin >> talkingAmt;
}
cout << "--- ";
/* Validate all the data entered by the user */
if (name.length() == 0)
{
    // if an error msg was already printed, ignore this error
    if (!showedAnError)
    
        cout << "You must enter a valid name." << endl;
    valid = false;
    showedAnError = true;
}

if (textingAnswer == YES)
{
    texting = true;
}
else if (textingAnswer != NO)
{
    // if an error msg was already printed, ignore this error
    if (!showedAnError)
    
        cout << "Your texting answer must be either Yes or No." << endl;
    valid = false;
    showedAnError = true;
}

if (webAnswer == YES)
{
    web = true;
}
else if (webAnswer != NO)
{
    // if an error msg was already printed, ignore this error
    if (!showedAnError)
    
        cout << "Your web answer must be either Yes or No." << endl;
    valid = false;
    showedAnError = true;
}

if (wifiAnswer == YES)
{
    wifi = true;
}
else if (wifiAnswer != NO)
{
    // if an error msg was already printed, ignore this error
    if (!showedAnError)
    
        cout << "Your wifi answer must be either Yes or No." << endl;
    valid = false;
    showedAnError = true;
}

if (talkingAmt < 0)
{
    // if an error msg was already printed, ignore this error
    if (!showedAnError)
    
        cout << "Your minutes talking must not be negative." << endl;
    valid = false;
    showedAnError = true;
}
if (valid)
{
    /* Determine the bill cost */
    cost = 30.00;
    if (texting) cost += 10;
    if (web) cost += 10;
    if (wifi) cost += 5;
    /* talking cost */
    cost += talkingAmt / 10;

    cout.precision(2);
    cout.setf(ios::showpoint);
    cout.setf(ios::fixed);
    cout << name << ", your charges this month are $" << cost << "." << endl;
}
return 0;

void promptForInput(string& name, string& texting, string& web, string& wifi, int& minutes)
{
    cout << "Customer Name? ";
    getline(cin, name);
    cout << "Want unlimited texting? ";
    getline(cin, texting);
    cout << "Want unlimited web? ";
    getline(cin, web);
    cout << "Want wifi hotspots? ";
    getline(cin, wifi);
    cout << "How many minutes talking did you use this month? ";
    cin >> minutes;
    return;
}

bool isValid(string name, string textingAnswer, string webAnswer, string wifiAnswer, int talkingAmt,
string& errorMessage)
{
    bool valid = true;
    bool foundAnError = false;
    const string YES = "Yes";
    const string NO = "No";

    /* Validate all the data entered by the user */
    if (name.length() == 0)
    {
        if (!foundAnError)
            errorMessage = "You must enter a valid name."
        valid = false;
        foundAnError = true;
    }

    if (textingAnswer == YES)
    {
        }
    else if (textingAnswer != NO)
    {
        if (foundAnError)
            errorMessage = "Your texting answer must be either Yes or No.";
    }
valid = false;
foundAnError = true;
}

if (webAnswer == YES)
{
}
else if (webAnswer != NO)
{
    if (!foundAnError)
        errorMessage = "Your web answer must be either Yes or No.";
    valid = false;
    foundAnError = true;
}

if (wifiAnswer == YES)
{
}
else if (wifiAnswer != NO)
{
    if (!foundAnError)
        errorMessage = "Your wifi answer must be either Yes or No.";
    valid = false;
    foundAnError = true;
}
if (talkingAmt < 0)
{
    if (!foundAnError)
        errorMessage = "Your minutes talking must not be negative.";
    valid = false;
    foundAnError = true;
}
return(valid);

bool wantsIt(string service)
{
    const string YES = "Yes";
    return(service == YES);
}

double calculateCost(string texting, string web, string wifi, int minutes)
{
    double cost(0.0);
    /* Determine the bill cost */
    cost = 30.00;
    if (wantsIt(texting)) cost += 10;
    if (wantsIt(web)) cost += 10;
    if (wantsIt(wifi)) cost += 5;
    /* talking cost */
    cost += minutes / 10;
    return(cost);
}
void showOutput(string name, double amount)
{
    cout.precision(2);
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout << name << ", your charges this month are $" << amount << "." << endl;
}

void showError(string errorMessage)
{
    cout << "--- " << errorMessage << endl;
}