Ranoa, Julius
CSC 121 001 Computer Science
22 September 2017 Friday

Part I. Review Questions and Exercises.
Chapter 6 Functions.
Qn. 1, 2, 3, 5, 7, 9, 10, 13, 14, 15, 20, 21, 23.

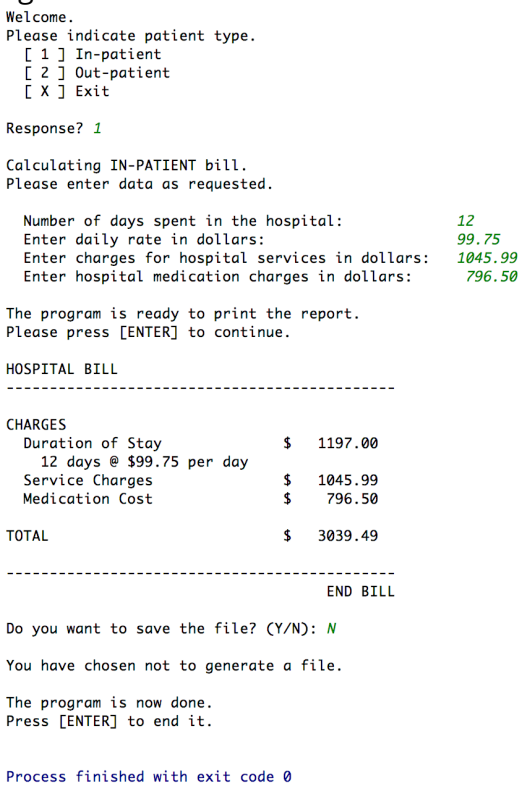
1. The function header is a part of function definition that shows the function name, return type, and parameter list.
2. If a function doesn't return a value, the word void will appear as its return type.
3. If a function `showValue` has the following header: `void showValue(int quantity)` you would use the statement `showValue(5)` to call it with argument 5.
5. Values that are sent into a function are called arguments.
7. When only a copy of an argument is passed to a function, it is said to be passed by value.
9. A(n) function prototype eliminates the need to place a function definition before all calls to the function.
10. Global variables are defined outside all functions and are accessible to any function within their scope.
13. If a function has a local variable with the same name as a global variable, only local variable can be seen by the function.
14. Static local variables retain their value between function calls.
15. The return statement causes a function to end immediately.
20. Reference variables are defined like regular variables, except there is a(n) & (ampersand) in front of the name.
21. Reference variables allow arguments to be passed by reference.
23. Two or more functions may have the same name, as long as their parameter lists are different.

Part II. Programming Challenge.
Page 399. Qn.16 Overloaded Hospital

Requirements:
1. Ask user if patient is either "inpatient" or "outpatient".
2. If patient is "inpatient", ask the user for the following information: (1) length of stay in days, (2) daily rate, (3) hospital service charges, (4) medication costs.
3. If patient is "outpatient", only ask for the following information: (3) hospital service charges, and (4) medication costs.
4. Validate that each input from point 2 and 3 is not less than zero. If it is, user input should be re-entered.
5. Use two overloaded functions to calculate the total charges, one for "inpatient" and the other for "outpatient". Return the total charge.

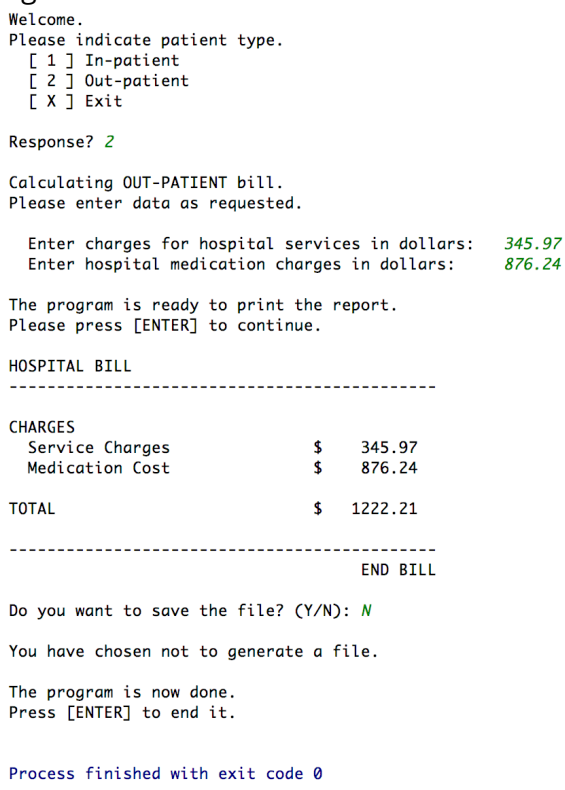
Screenshot of Runtime.

Figure 1. With INPATIENT data



```
Welcome.
Please indicate patient type.
  [ 1 ] In-patient
  [ 2 ] Out-patient
  [ X ] Exit

Response? 1

Calculating IN-PATIENT bill.
Please enter data as requested.

  Number of days spent in the hospital:            12
  Enter daily rate in dollars:                     99.75
  Enter charges for hospital services in dollars:  1045.99
  Enter hospital medication charges in dollars:    796.50

The program is ready to print the report.
Please press [ENTER] to continue.

HOSPITAL BILL
-----------------------------------------------

CHARGES
  Duration of Stay             $   1197.00
    12 days @ $99.75 per day
  Service Charges              $   1045.99
  Medication Cost              $    796.50

TOTAL                          $   3039.49

-----------------------------------------------
                                    END BILL

Do you want to save the file? (Y/N): N

You have chosen not to generate a file.

The program is now done.
Press [ENTER] to end it.


Process finished with exit code 0
```

Figure 2. With OUTPATIENT data



```
Welcome.
Please indicate patient type.
  [ 1 ] In-patient
  [ 2 ] Out-patient
  [ X ] Exit

Response? 2

Calculating OUT-PATIENT bill.
Please enter data as requested.

  Enter charges for hospital services in dollars:  345.97
  Enter hospital medication charges in dollars:    876.24

The program is ready to print the report.
Please press [ENTER] to continue.

HOSPITAL BILL
-----------------------------------------------

CHARGES
  Service Charges              $    345.97
  Medication Cost              $    876.24

TOTAL                          $   1222.21

-----------------------------------------------
                                    END BILL

Do you want to save the file? (Y/N): N

You have chosen not to generate a file.

The program is now done.
Press [ENTER] to end it.


Process finished with exit code 0
```

Project Files.

(1) main.cpp
(2) HospitalBill.h
(3) HospitalBill.cpp

Source code is included on Pages 3 – 10.
For the actual files, visit the following link:

https://github.com/TheLoneWoof1102/FA17_CSC121001/tree/master/Source%20Code/Homework-Ch6.Qn16

It's a link to a folder in a public repository in GitHub that contains all my work for this class.

main.cpp

```cpp
#include "HospitalBill.h"

int main() {
    HospitalBill h;
    h.runInterface();
}
```

HospitalBill.h

```cpp
#include <string>

class HospitalBill {

private:
    int num_days;
    double daily_rate;
    double service_charges;
    double medication_charges;
    double total_charges;

public:
    // Constructor
    HospitalBill() {
        num_days = daily_rate = service_charges = medication_charges = 0;
    }

    // Validation Functions
    bool isPositive(int);
    bool isPositive(double);

    // Calculation Methods
    double getTotalCharges(int, double, double, double);
    double getTotalCharges(double, double);

    // Interface Methods
    void runInterface();
    char getUserOption();
    bool askForNumbers(double&); // Returns true if successful. False if not.
    bool askForIntegers(int&);
    std::string printTextBlock(std::string, int, bool = true);
    std::string printCurrency(std::string, double, int, int = 2);
    std::string makeReport();
    void printReport();

    // File Making Methods
    bool makeFile(std::string, std::string);

    // Calculation Methods
    void getInpatientData();
    void getOutpatientData();
};
```

HospitalBill.cpp

```cpp
#include <iostream>
#include <iomanip>
#include <sstream>
#include <fstream>
#include "HospitalBill.h"
using namespace std;

/*
 * This function contains the general branching logic
 * of the program. It is also the only function that
 * is called in main()
 *
 * */
void HospitalBill::runInterface() {
    switch( getUserOption() ) {
        case '1':
            getInpatientData();
            printReport();
            break;
        case '2':
            getOutpatientData();
            printReport();
            break;
    }
    cout << "The program is now done." << endl;
    cout << "Press [ENTER] to end it." << endl;
    cin.get();
}

/*
 * This function presents a list of key-description
 * pairs. The keys are single characters and the user
 * is prompted to choose one of the keys.
 *
 * It also validates the keys and re-prompts the users
 * for the key if invalid.
 *
 * Also, each line submitted is considered separate
 * input. This is done so that invalid input in one
 * attempt does not bleed over to the next.
 *
 * Returns valid key from input.
 *
*/
char HospitalBill::getUserOption() {
    // Key-Description Pairs
    // All keys must be uppercase, if alpha.
    const std::string opts[][2] = {
            { "1", "In-patient" },
            { "2", "Out-patient" },
            { "X", "Exit" }
    };
    string raw_input;       // Container for user input
    bool input_is_valid;    // Flag

    cout << "Welcome." << endl;
    cout << "Please indicate patient type." << endl;
```

HospitalBill.cpp – cont'd.

```
    // Prints out all key-description pairs in the opts array
    for (int i = 0; i < sizeof opts / sizeof opts[0]; i++) {
        // Prints it in this format: [ Key ] Description
        cout << "  [ " <<  opts[i][0] << " ] " << opts[i][1] << endl;
    }
    cout << endl;

    cout << "Response? ";
    getline(cin, raw_input);

    // Flag is by default false. Inside the loop,
    // the input is validated first, then if the input is invalid,
    // it prompts the user again for a key.
    input_is_valid = false;
    while (!input_is_valid) {
        // If the input key is lowercase,
        // make it uppercase.
        raw_input = string(1, toupper(raw_input[0]));
        // Check if it is in the valid options
        for (int i = 0; i < sizeof opts / sizeof opts[0]; i++) {
            if (raw_input == opts[i][0]) {
                input_is_valid = true;
                break;
            }
        }
        if (!input_is_valid) {
            cout << "Invalid input. Please try again: ";
            getline(cin, raw_input);
        }
    }

    cout << endl;
    return toupper(raw_input[0]);
}

/*
 * This function prompts the user for inpatient data.
 * If input is invalid, informs the user and re-prompts
 * them for data.
 *
 * After all in-patient related data is in,
 * the total charge is calculated.
 * All input data is saved inside the object.
 *
 * */
void HospitalBill::getInpatientData() {
    int left_col_width = 52;

    cout << "Calculating IN-PATIENT bill." << endl;
    cout << "Please enter data as requested." << endl;
    cout << endl;

    cout << printTextBlock("  Number of days spent in the hospital:", left_col_width);
    while (!askForIntegers(num_days)) {
        cout << printTextBlock("    Invalid input. Try again:", left_col_width);
    }
```

HospitalBill.cpp – cont'd.

```
    cout << printTextBlock("  Enter daily rate in dollars: ", left_col_width);
    while (!askForNumbers(daily_rate)) {
        cout << printTextBlock("    Invalid input. Try again:", left_col_width);
    }

    cout << printTextBlock("  Enter charges for hospital services in dollars: ", left_col_width);
    while (!askForNumbers(service_charges)) {
        cout << printTextBlock("    Invalid input. Try again:", left_col_width);
    }

    cout << printTextBlock("  Enter hospital medication charges in dollars: ", left_col_width);
    while (!askForNumbers(medication_charges)) {
        cout << printTextBlock("    Invalid input. Try again:", left_col_width);
    }

    cout << endl;
    total_charges = getTotalCharges(num_days, daily_rate, service_charges, medication_charges);
    return;
}

/*
 * This function prompts the user for outpatient data.
 * If input is invalid, informs the user and re-prompts
 * them for data.
 *
 * After all out-patient related data is in,
 * the total charge is calculated.
 * All input data is saved inside the object.
 *
 * */
void HospitalBill::getOutpatientData() {
    int left_col_width = 52;

    cout << "Calculating OUT-PATIENT bill." << endl;
    cout << "Please enter data as requested." << endl;
    cout << endl;

    cout << printTextBlock("  Enter charges for hospital services in dollars: ", left_col_width);
    while (!askForNumbers(service_charges)) {
        cout << printTextBlock("    Invalid input. Try again:", left_col_width);
    }

    cout << printTextBlock("  Enter hospital medication charges in dollars: ", left_col_width);
    while (!askForNumbers(medication_charges)) {
        cout << printTextBlock("    Invalid input. Try again:", left_col_width);
    }

    cout << endl;
    total_charges = getTotalCharges(service_charges, medication_charges);
    return;
}
```

HospitalBill.cpp – cont'd.

```
/*
 * This function returns a string object containing
 * the report.
 *
 * It uses a string stream to replace cout,
 * and evaluates that stream to produce the string.
 *
 * */
string HospitalBill::makeReport() {
    stringstream ss;

    int width_titles = 30,
        width_val = 10,
        width_margins = 2,
        width_total = width_titles + 1 + width_val + (width_margins * 2);

    string margin, border;
    margin.assign(width_margins, ' ');
    border.assign(width_total, '-');

    ss << "HOSPITAL BILL" << endl;
    ss << border << endl;
    ss << endl;

    ss << "CHARGES" << endl;
    if (num_days > 0 && daily_rate > 0) {
        ss << margin
           << printTextBlock("Duration of Stay", width_titles)
           << printCurrency("$", (num_days * daily_rate), width_val)
           << endl;

        ss << margin << margin
           << to_string(num_days) << " days @ "
           << printCurrency("$", daily_rate, 0)
           << " per day" << endl;
    }
    if (service_charges > 0) {
        ss << margin
           << printTextBlock("Service Charges", width_titles)
           << printCurrency("$", service_charges, width_val)
           << endl;
    }
    if (medication_charges > 0) {
        ss << margin
           << printTextBlock("Medication Cost", width_titles)
           << printCurrency("$", medication_charges, width_val)
           << endl;
    }
    ss << endl
       << printTextBlock("TOTAL", width_titles + width_margins)
       << printCurrency("$", total_charges, width_val)
       << endl;

    ss << endl << border << endl;
    ss << printTextBlock("END BILL", width_total, false);
    ss << endl << endl;

    return ss.str();
}
```

HospitalBill.cpp – cont'd

```
/*
 * Prompts the user for reporting.
 * Prints the report created from makeReport()
 *
 * Then, prompts the user if they want a file
 * containing the report to be created.
 *
 * */
void HospitalBill::printReport() {
    string report, raw_input;
    bool input_is_valid, make_file = false;

    cout << "The program is ready to print the report." << endl;
    cout << "Please press [ENTER] to continue." << endl;
    getline(cin, report);

    report = makeReport();
    cout << report;

    cout << "Do you want to save the file? (Y/N): ";
    getline(cin, raw_input);

    input_is_valid = false;
    while (!input_is_valid) {
        // Check if it is in the valid options
        switch( raw_input[0] ) {
            case 'Y':
            case 'y':
                make_file = true;
            case 'N':
            case 'n':
                input_is_valid = true;
        }
        if (!input_is_valid) {
            cout << "Invalid input. Please try again: ";
            getline(cin, raw_input);
        }
    }

    cout << endl;
    if (make_file) {
        cout << "Creating text file. Please wait. " << endl;
        if (makeFile("Hospital Charge Report.txt", report)) {
            cout << "Report file is created." << endl;
        } else {
            cout << "We're sorry. Something went wrong." << endl;
        }
    } else {
        cout << "You have chosen not to generate a file." << endl;
    }

    cout << endl;
    return;
}
```

HospitalBill.cpp – cont'd.

```
/*
 * Makes a file given a filename and some content.
 * Handles all file-related operations.
 * Returns a flag on whether file creation was successful.
 *
 * */
bool HospitalBill::makeFile(string filename, string content) {
    ofstream report_file;

    try {
        report_file.open(filename);
        report_file << content;
        report_file.close();
    } catch (exception e) {
        return false;
    }

    return true;
}

/*
 * Calculates the total charges for in-patient data using the ff. formula:
 * Total = (Days Stayed * Daily Rate) + Service + Medicine.
 *
 * */
double HospitalBill::getTotalCharges(int nd, double dr, double sc, double mc) {
    return (nd * dr) + sc + mc;
}

/*
 * Calculates total charges for out-patient data using the ff. formula.
 * Total = Service + Medicine
 *
 * */
double HospitalBill::getTotalCharges(double sc, double mc) {
    return sc + mc;
}

/*
 * Accepts a string and returns it with width and alignment formatting.
 * Used so much in the program, I've had to create a function for it.
 *
 * */
string HospitalBill::printTextBlock(string s, int width, bool left_align) {
    stringstream ss;
    ss << setw(width) << ((left_align) ? left : right ) << s;
    return ss.str();
};
```

HospitalBill.cpp – cont'd.

```
/*
 * Accepts a double variable and returns a string with the number along
 * with some width and precision formatting.
 *
 * */
string HospitalBill::printCurrency(std::string sym, double val, int width, int precision) {
    stringstream ss;
    ss << sym;
    ss << fixed << showpoint << setprecision(precision);
    ss << setw(width) << right << val;

    return ss.str();
}

/*
 * This function prompts the user for a number but does it
 * via an intermediary string variable.
 *
 * This is done so that if a user inputs troublesome data,
 * it would NOT bleed over the next input prompt.
 *
 * Accepts a reference to a variable,
 * saves the extracted double value to it,
 * and returns true if all if fine. False if not.
 *
 * */
bool HospitalBill::askForNumbers(double &data) {
    string raw_input;
    getline(cin, raw_input);

    try {
        data = stod(raw_input);
        return true;
    } catch (exception e) {
        return false;
    }
};

/*
 * See HospitalBill:askForNumbers().
 * Only this function is for integers.
 *
 * */
bool HospitalBill::askForIntegers(int &data) {
    string raw_input;
    getline(cin, raw_input);

    try {
        data = stoi(raw_input);
        return true;
    } catch (exception e) {
        return false;
    }
};
```

// End of *HospitalBill.cpp*