### SWE20004 Technical Software Development Semester 2 2019

# Assignment 2 report

## Name: Jake Scott Student ID: 102581840

## Lab class: Wednesday / 12:30 PM / BA601

**Due Date: Friday 11th September 2019 at 11:59pm**

**Date Submitted: Friday 11th September 2019 at 01:00am**

## Assignment Title: 1

### Program description

The program is used to validate credit card numbers using Luhn’s algorithm to validate correct credit card numbers. Once a user has entered their credit card number (with spaces in between numbers) then a -1 integer is used to signify the end of the credit card number. Then the algorithm is applied depending on weather the credit card is valid/invalid it is printed to the screen.

### Inputs and Outputs

The inputs and outputs for this program are described in Table 1.

**Table 1. Data dictionary:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Data to be stored** | **Sample data** | **Type of data** | **C++ type** | **Input method** | **In / Out** | **Variable name** |
| Integers | 7 | Int | Int array | Cin | input | creditCard |
| Integers | 8 | Int | Int | creditCard | n/a | checkSumPostion |
| Integers | 2 | Int | Int array | creditCard | n/a | sumOne |
| Integers | 3 | int | Int | sumOne | n/a | First\_digit |
| Integers | 4 | int | int | sumOne | n/a | Last\_digit |
| Integers | 5 | int | int | sumOne | n/a | sumOneTotal |
| Integers | 6 | Int | Int array | creditCard | n/a | sumTwo |
| Integers | 7 | Int | Int | sumTwo | n/a | sumTwoTotal |
| Integers | 8 | Int | Int | creditCard | n/a | checkNumber |
| Integers | 9 | Int | Int | sumOne | Output | totalOne |
| Integers | 1 | Int | Int | sumTwo | Output | totalTwo |
| Integers | 2 | Int | Int | totalOne, totalTwo | n/a | total |
| Integers | 3 | Int | Int | total | output | checkSumDigit |

### 3. Algorithm

Program steps:

1. Prompt the user for credit card numbers (followed by a -1 integer)
2. Store credit card numbers in an array
3. Display users input (whitespace and -1 integer removed)
4. Calculate sum1 and print subsequent integers
5. Calculate sum2 and print subsequent integers
6. Calculate total for sum1 and print to screen
7. Calculate total for sum2 and print to screen
8. Calculate and print to screen checkSumNumber
9. If checkSumNumber and last digit of credit card are the same program deems credit card valid, if not print invalid

Encoding & Decoding logic:

Sum 1 (represented by f(x)):

If sum1 is greater than 10:

Sum 2 (represented by h(x)):

A close up of a logo

Description automatically generatedFlow Chart:

### Source code:

/\*

Filename: 102581840.cpp

Purpose: Assignment Luhn Algorithm Credit Card Validity Checker

Subject: SWE20004

Author: Jake Scott (120581840)

Date Last Modifed: 1/10/2019

\*/

#include <iostream>

#include <cmath>

using namespace std;

int checkSumPostion = 0;

int chkNum(int sumOne[]) {

for (int i = 0; i < 20; i++) {

if (sumOne[i] == -1) {

int checkNumber = sumOne[i-1];

checkSumPostion = i - 1;

return checkNumber;

}

}

}

int sum1(int sumOne[]) {

int sumOneTotal = 0;

cout << "" << endl;

cout << "Numbers for sum 1 are: ";

for (int i = 0; i < checkSumPostion; i+=2) {

if (sumOne[i] < 0) {

break;

}

cout << sumOne[i] << " ";

sumOne[i] = sumOne[i] + sumOne[i];

if (sumOne[i] >= 10) {

int first\_digit = sumOne[i] / 10;

int last\_digit = sumOne[i] % 10;

sumOne[i] = first\_digit + last\_digit;

}

sumOneTotal += sumOne[i];

}

cout << "" << endl;

return sumOneTotal;

}

int sum2(int sumTwo[]) {

int sumTwoTotal = 0;

cout << "Numbers for sum 2 are: ";

for (int i = 1; i < checkSumPostion; i+=2) {

if (sumTwo[i] < 0) {

break;

}

cout << sumTwo[i] << " ";

sumTwoTotal += sumTwo[i];

}

return sumTwoTotal;

}

int main()

{

int creditCard[20];

cout << "Please enter credit card numbers:" << endl;

for (int i = 0; i < 20; i++) {

cin >> creditCard[i];

if (creditCard[i] == -1) {

break;

}

}

cout << "Credit card number is: ";

for (int j = 0; j < 20; j++) {

if (creditCard[j] == -1 ) {

break;

}

cout << creditCard[j];

}

int checkNumber = chkNum(creditCard);

int totalOne = sum1(creditCard);

int totalTwo = sum2(creditCard);

int total = totalOne + totalTwo;

total = total \* 9;

int checkSumDigit = total % 10;

cout << "" << endl;

cout << "" << endl;

cout << "Sum 1 is: " << totalOne << endl;

cout << "Sum 2 is: " << totalTwo << endl;

cout << "" << endl;

cout << "Check Number is: " << checkNumber << endl;

cout << "" << endl;

if (checkSumDigit == checkNumber) {

cout << "Check sum is " << checkSumDigit << " and the last digit " << checkNumber << " are the same: Valid credit card number" << endl;

}

else {

cout << "Check sum is " << checkSumDigit << " and the last digit " << checkNumber << " are not the same: Invalid credit card number" << endl;

}

return 0;

}

### Screenshots showing working program (Show all possible outcome):

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Task 5.3

### Program description

Program generates a number between 1 and 1000, and then the user attempts to guess this number. If the number is higher than the generated number the program prints a “Too high” response, if the number is too low the program prints a “Too low” response. Once the number has been guessed the program prompts the user if they would like to play again, if yes, the program generates a new number, if not program terminates.

### Source Code

/\*

Filename: Task 5.3.cpp

Purpose: Task 5.3 "Guess the Number" game

Subject: SWE20004

Author: Jake Scott (120581840)

Date Last Modifed: 10/10/2019

\*/

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

int main()

{

int number, input;

string loop;

bool i = true;

srand(time(NULL));

number = 1 + rand() % 1000;

cout << "I have a number between 1 and 1000." << endl;

cout << "Can you guess my number?" << endl;

cout << "Please type your first guess." << endl;

while (i) {

cin >> input;

if (input > number) {

cout << "Too high. Try Again." << endl;

}

else if (input < number) {

cout << "Too low. Try Again." << endl;

}

else if (input == number) {

cout << "Excellent! You guessed the number!" << endl;

cout << "Would you like to play again (yes or no)?" << endl;

cin >> loop;

if (loop == "no") {

i = false;

}

}

}

return 0;

}

### Screenshots showing working program

A screenshot of a computer

Description automatically generated

## Task 5.4

### Program description

Program prompts the user for a wager, the default bank value is $1000. Once the user has entered their wager (Must be lower or equal to the bank value). Program generates two dice numbers (numbers 1 to 6). Then the sum of the two dice are calculated. If the number is 7 or 11 the player loses. If the total is equal to 4, 5, 6, 8, 9, 10. Then the program continues to roll the dice until the dice total equal the original value, if one of the rolled totals is equal to 7 then the player loses. If however the player wins their wager is doubled and placed into the bank for future games.

### Source Code

/\*

Filename: Task 5.4.cpp

Purpose: Task 5.4, Rolling Dice Game

Subject: SWE20004

Author: Jake Scott (120581840)

Date Last Modifed: 10/10/2019

\*/

#include <iostream>

#include <stdlib.h>

#include <time.h>

using namespace std;

int diceRoll() {

return rand() % 6 + 1;

}

int main()

{

bool i = true;

int bankBalance = 1000;

while(i) {

bool j = true;

int bet;

srand(time(NULL));

cout << "You have $" << bankBalance << " in the bank." << endl;

cout << "Place your wager: ";

cin >> bet;

bankBalance = bankBalance - bet;

int diceOne = diceRoll();

int diceTwo = diceRoll();

int diceTotal = diceOne + diceTwo;

if (diceTotal == 2 || diceTotal == 3 || diceTotal == 12) {

cout << "Player Loses, better luck next time!" << endl;

}

else if (diceTotal == 7 || diceTotal == 11) {

cout << "Player Wins!!" << endl;

bankBalance = bet \* 2 + bankBalance;

}

else {

cout << "Player Rolled " << diceOne << " + " << diceTwo << " = " << diceTotal << endl;

cout << "Point is " << diceTotal << endl;

cout << "Way too lucky! THose dice have to be loaded!" << endl;

cout << "You're up big. Now's the time to cash in your chips!" << endl;

while(j) {

int diceThree = diceRoll();

int diceFour = diceRoll();

int subDiceTotal = diceThree + diceFour;

cout << "Player Rolled " << diceThree << " + " << diceFour << " = " << subDiceTotal << endl;

if (subDiceTotal == 7) {

cout << "Player Loses!" << endl;

j = false;

}

else if (subDiceTotal == diceTotal) {

cout << "Player Wins!!" << endl;

bankBalance = bet \* 2 + bankBalance;

j = false;

}

}

}

string userInput;

cout << "Would you like to try again( y/n)?" << endl;

cin >> userInput;

if (userInput == "n") {

cout << "Thank you for playing!" << endl;

i = false;

break;

}

}

return 0;

}

### Screenshots showing working program

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Task 7.3

### Program description

Program prompts user for 20 inputs of alphabetic characters only, while inputting if the user enters a letter that has already been entered before the program prints the letter followed by “is a duplicate letter.”. Then the program prints the original array, followed by the array sorted alphabetically.

### Source Code

/\*

Filename: Task 7.3.cpp

Purpose: Task 7.3, Sorting letter arrays

Subject: SWE20004

Author: Jake Scott (120581840)

Date Last Modifed: 10/10/2019

\*/

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <cctype>

#include <clocale>

#include <bits/stdc++.h>

using namespace std;

int main()

{

cout << "Enter 20 alphabets" << endl;

int i,k,j,x, check;

bool z = true;

unsigned char array[20];

for (i=0; i < 20; i++) {

check = 0;

cin >> array[i];

if (!(bool)std::isalpha(array[i])) {

while (z) {

cout << array[i] << "Is not a letter" << endl;

cin >> array[i];

if ((bool)std::isalpha(array[i])) {

z = false;

}

}

}

for (k=0; k < i; k++) {

if (array[i] == array[k])

check = 1;

}

if (check == 1) {

cout << array[i] << " is a duplicate letter" << endl;

}

}

cout << "" << endl;

cout << "The Original array is: " << endl;

for (j=0; j < 20; j++) {

cout << array[j];

}

cout << "" << endl;

cout << "The Sorted array is: " << endl;

int n = sizeof(array)/sizeof(array[0]);

sort(array, array+n);

for (x=0; x < 20; x++) {

cout << array[x];

}

return 0;

}

### Screenshots showing working program

A screenshot of a computer

Description automatically generated

## Task 7.4

### Program description

The program rolls two dice (numbers 1 to 6) 50,000 times and record the combination of dice rolls and stores them in a two-dimensional array. It then prints out how many times each combination of rolls occurred within the 50,000 rolls. After that the program then prints out all diagonal elements (rolls 1 and 1, 2 and 2, 3 and 3, 4 and 4, 5 and 5, 6 and 6). Then calculates a total for these rolls. If the total of rolls is less than 8350 then the player loses then game. However, if the total rolls are greater than 8350 the player wins the game.

### Source Code

/\*

Filename: Task 7.4.cpp

Purpose: Task 7.4, Dice rolling game

Subject: SWE20004

Author: Jake Scott (120581840)

Date Last Modifed: 10/10/2019

\*/

#include <iostream>

#include <stdlib.h>

#include <time.h>

using namespace std;

int diceRoll() {

return rand() % 6 + 1;

}

int main()

{

srand(time(NULL));

int roll[7][7] = {0};

for (int i = 0; i < 50000; i++) {

roll[diceRoll()][diceRoll()] += 1;

}

for (int j = 1; j < 7; j++) {

cout << "" << endl;

for (int k = 1; k < 7; k++) {

cout << "|" << roll[j][k] << " | ";

}

}

cout << "" << endl;

cout << "" << endl;

cout << "The sum of the diagonal elements is: " << endl;

int total = 0;

for (int x = 1; x < 7; x++) {

cout << roll[x][x] << endl;

total = roll[x][x] + total;

}

cout << "------" << endl;

cout << total << endl;

if (total > 8350) {

cout << total << " is greater than 8350, the player won the game!!" << endl;

}

else {

cout << total << " is less than 8350, the player lost the game :(" << endl;

}

return 0;

}

### Screenshots showing working program

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated