Rajshahi University of Engineering & Technology

CSE 2102: Sessional Based on CSE 2101

Lab Report 01

Dated: 17.02.18

Submitted to

Rizoan Toufiq

Assistant Professor, Dept. of CSE

&

Instructor, CSE 2102

Submitted by

Fuad Al Abir

Roll: 1603021

Section: A

Dept. of CSE

**Experiment No. 01**

**Name of the Experiment**: The Foundations: Logic and Proof

**1. EXPERIMENT [ 1 ]**

Generate Truth Table for the propositional value of p and q find the negation of p and q, conjunction (p ∧ q), disjunction (p ∨ q), Exclusive OR (p ⊕ q), Conditional Statement (p -> q), Bi-conditional (p ↔ q) Statement.

**THEORY:** Truth table for the basic propositional value of two variables, p and q is generated using an array of Boolean values and various functions with corresponding tasks. For each input, the second character was taken on account of calling the corresponding function, generate the truth values of different combinations and print them.

#include <iostream>

#include <stdio.h>

#include <cstring>

using namespace std;

// note to the users...

// symbols used in this program...

// ^ for conjunction

// v for disjunction

// o for Exclusive OR

// > for conditional

// } for bi-conditional

// ! for negation

//

// exit for exit!

bool p[4] = {true, true, false, false};

bool q[4] = {true, false, true, false};

bool r[4], b[2];

conjunction(bool \*p, bool \*q)

{

for(int i = 0; i < 4; i++)

{

if(p[i] == true && q[i] == true) r[i] = true;

else r[i] = false;

}

}

disjunction(bool \*p, bool \*q)

{

for(int i = 0; i < 4; i++)

{

if(p[i] == false && q[i] == false) r[i] = false;

else r[i] = true;

}

}

void XOR(bool \*p, bool \*q)

{

for(int i = 0; i < 4; i++)

{

if((p[i] == true && q[i] == true) || (p[i] == false && q[i] == false)) r[i] = false;

else r[i] = true;

}

}

void conditional(bool \*p, bool \*q)

{

for(int i = 0; i < 4; i++)

{

if((p[i] == true && q[i] == true) || p[i] == false) r[i] = true;

else r[i] = false;

}

}

void biconditional(bool \*p, bool \*q)

{

for(int i = 0; i < 4; i++)

{

if((p[i] == true && q[i] == true) || (p[i] == false && q[i] == false)) r[i] = true;

else r[i] = false;

}

}

void negation(bool \*p)

{

r[0] = p[0] ? false : true;

r[1] = p[3] ? false : true;

}

void pnprint(bool \*p, bool \*r, char \*str)

{

printf("\tp\t|\t!p\t\n----------------------------------\n");

printf("\t%s\t\t%s\n", p[0] ? "true" : "false", r[0] ? "true" : "false");

printf("\t%s\t\t%s\n", p[3] ? "true" : "false", r[1] ? "true" : "false");

}

void qnprint(bool \*q, bool \*r, char \*str)

{

printf("\tq\t|\t!q\t\n----------------------------------\n");

printf("\t%s\t\t%s\n", q[0] ? "true" : "false", r[0] ? "true" : "false");

printf("\t%s\t\t%s\n", q[3] ? "true" : "false", r[1] ? "true" : "false");

}

void print(bool \*p, bool \*q, bool \*r, char \*str)

{

cout << "\tp\t|\tq\t|\tp " << str[1] << " q\t\n-----------------------------------------------------\n";

for(int i = 0; i < 4; i++)

{

printf("\t%s\t\t%s\t\t%s\n", p[i] ? "true" : "false", q[i] ? "true" : "false", r[i] ? "true" : "false");

}

}

int main()

{

char str[5];

while(true)

{

cin >> str;

//char ch;

//solving space ambiguity

//if(strlen(str) <= 3) ch = str[1];

//else if(strlen(str) >= 4) ch = str[2];

switch(str[1])

{

case '^': conjunction(p, q); print(p, q, r, str); break;

case 'v': disjunction(p, q); print(p, q, r, str); break;

case 'o': XOR(p, q); print(p, q, r, str); break;

case '>': conditional(p, q); print(p, q, r, str); break;

case '}': biconditional(p, q); print(p, q, r, str); break;

case 'p': negation(p); pnprint(p, r, str); break;

case 'q': negation(q); qnprint(q, r, str); break;

case 'x': return 0;

default: break;

}

}

return 0;

}

TERMINAL:

Musafeer@DESKTOP-OGK5MQH MINGW64 ~/Desktop/s t u d y/cse 2102/lab 1/code

$ g++ mod1.1.cpp

Musafeer@DESKTOP-OGK5MQH MINGW64 ~/Desktop/s t u d y/cse 2102/lab 1/code

$ ./a.exe

p^p

p | q | p ^ q

-----------------------------------------------------

true true true

true false false

false true false

false false false

pvp

p | q | p v q

-----------------------------------------------------

true true true

true false true

false true true

false false false

pop

p | q | p o q

-----------------------------------------------------

true true false

true false true

false true true

false false false

p>q

p | q | p > q

-----------------------------------------------------

true true true

true false false

false true true

false false true

p}q

p | q | p } q

-----------------------------------------------------

true true true

true false false

false true false

false false true

!p

p | !p

----------------------------------

true false

false true

!q

q | !q

----------------------------------

true false

false true

exit

**DISCUSSION**: Though it works quiet well, but this program has some ambiguity with the spaces while inputting the basic proposition string. If the input format and the symbols are not according to stated as above, this program may fail or won’t be able to show the expected truth table.

**2. EXPERIMENT [ 3 ]**

Given two bit strings of length n, find the bitwise AND, bitwise OR and bitwise XOR of these strings.

**THEORY:** The size of the string was taken as user input first and then the two strings respectively. Then the bitwise operations was done by the corresponding functions. A menu program is created to take the user input to perform the operations.

#include <iostream>

#include <string>

using namespace std;

AND(char \*str1, char \*str2, int size)

{

cout << "Bitwise AND of two bit string " << str1 << " and " <<str2<<": ";

for(int i = 0; i < size; i++)

{

if(str1[i] == '1' && str2[i] == '1')

{

cout << "1";

}

else cout << "0";

}

cout << "\n\n";

}

OR(char \*str1, char \*str2, int size)

{

cout << "Bitwise OR of two bit string " << str1 << " and " <<str2<< ": ";

for(int i = 0; i < size; i++)

{

if(str1[i] == '0' && str2[i] == '0')

{

cout << "0";

}

else cout << "1";

}

cout << "\n\n";

}

XOR(char \*str1, char \*str2, int size)

{

cout << "Bitwise XOR of two bit string " << str1 << " and " << str2 << ": " ;

for(int i = 0; i < size; i++)

{

if(str1[i] == str2[i])

{

cout << "0";

}

else cout << "1";

}

cout << "\n\n";

}

int main()

{

int n;

cout << "Enter the string size: ";

cin >> n;

char str1[n], str2[n];

cout << "Enter the two bit strings of size " << n << "\nBit String 1: ";

cin >> str1;

cout << "Bit String 2: ";

cin >> str2;

cout << "\n";

while(1)

{

cout << "Enter your choice:\n\t1. Bitwise AND\n\t2. Bitwise OR\n\t3. Bitwise XOR\n\t4. Exit\n\n";

int ch;

cin >> ch;

switch(ch)

{

case 1: AND(str1, str2, n); break;

case 2: OR(str1, str2, n); break;

case 3: XOR(str1, str2, n); break;

default: return 0;

}

}

return 0;

}

TERMINAL:

Musafeer@DESKTOP-OGK5MQH MINGW64 ~/desktop/s t u d y/cse 2102/lab 1/code

$ g++ mod1.3.cpp

Musafeer@DESKTOP-OGK5MQH MINGW64 ~/desktop/s t u d y/cse 2102/lab 1/code

$ ./a.exe

Enter the string size: 4

Enter the two bit strings of size 4

Bit String 1: 1101

Bit String 2: 0011

Enter your choice:

1. Bitwise AND

2. Bitwise OR

3. Bitwise XOR

4. Exit

1

Bitwise AND of two bit string 1101 and 0011: 0001

Enter your choice:

1. Bitwise AND

2. Bitwise OR

3. Bitwise XOR

4. Exit

2

Bitwise OR of two bit string 1101 and 0011: 1111

Enter your choice:

1. Bitwise AND

2. Bitwise OR

3. Bitwise XOR

4. Exit

3

Bitwise XOR of two bit string 1101 and 0011: 1110

Enter your choice:

1. Bitwise AND

2. Bitwise OR

3. Bitwise XOR

4. Exit

4

**3. EXPERIMENT [8]**

Find the pair of (x, y) where x2 + y2 = z2, z is given, x, y and z are integers. [Note: could you check x3 + y3 = z3, have a solution where x, y and z are integers?]

**THEORY:** The problem is to find the Pythagorean triplets, (x, y, z is integer: x2 + y2 = z2) where z is given as user input. After getting the input z, a loop was created to find such x and y where x is the loop variable and y is being checked if it is an integer or not while this relation is true, y2 = z2 – x2. If all the factors are true, then x, y, z is a Pythagorean triplet. The solution is printed out then.

This algorithm also prevents printing the conjugate solution pair.

#include <stdio.h>

#include <math.h>

int main()

{

int x, y, z, flag = 0;

// scanning z as user input

scanf("%d", &z);

// iteration loop x = 2 to less than z

for(x = 2; x < z; x++)

{

int tempY;

// preventing printing the pair

// it's conjugate is printed before

if(y == tempY)

{

continue;

}

// storing the values of square root

// of y in two types, float and int

float yfs = z\*z - x\*x;

float yf = sqrt(yfs);

y = sqrt(yfs);

// checking if y is an integer

// then y and corresponding x pair is a solution

if(y == yf)

{

printf("(x, y): (%d, %d)\n", x, y);

tempY = y;

flag = 1;

}

}

// flag to check if any solution is not found...

if(flag == 0)

{

printf("No Possible Solition.\nTry again...\n");

main();

}

char ch;

scanf("%s", &ch);

return 0;

}

INPUT:

5

OUTPUT:

(x, y): (3, 4)

INPUT:

10

OUTPUT:

(x, y): (6, 8)

**DISCUSSION**: It’s not possible to find any integer solution for the equation x3 + y3 = z3, where as it’s not possible to find any integer solution when those power goes greater than 2. There is infinite solution for 1 and 2, but not for the aboves. This is a famous theorem in number theory given by the famous mathematician Pierre de Fermat in 1637, though it was not proved then. In 1994, the first successful prove of this theorem was released by Andrew Wiles.