#include <stdio.h>

#include <iostream>

using namespace std;

int bitOr(int a, int b){

int kq = ~((~a) & (~b));

return kq;

}

int negative (int x){

int kq = ((~x) + 1);

return kq;

}

int flipBit (int x, int n){

int kq = x ^ (1 << n);

return kq;

}

int get32bit(int x){

int kq = x & 0xffffffff;

return kq;

}

unsigned int divpw2(unsigned int x, int n){

int kq = x << negative(n);

return kq;

}

int isSameSign(int x, int y) {

int temp = x ^ y;

int sign = temp >> 31;

return sign + 1;

}

int isPositive(int x) {

return negative((negative(x) >> 31));

}

int isLess2n(int x, int n) {

int value2n = 2 << n + negative(1);

int diff = x + negative(value2n);

return negative((diff >> 31));

}

int getnbit(int x, int n){

int check = isLess2n (n, 5);

int kq = x & (0xffffffff >> ( ((n ^ 31) + 1) & (0b1111100000>>check>>check>>check>>check>>check) ) );

return kq;

}

int is16x(int x) {

bool check = x << 28;

return negative(check + negative(1));

}

void PrintBits(unsigned int x) {

int i;

for (i = 8 \* sizeof(x)-1; i >= 0; i--) {

(x & (1 << i)) ? putchar('1') : putchar('0');

if (i % 4 == 0){

cout << " ";

}

}

printf("\n");

}

void PrintBitsOfByte(unsigned int x) {

int i;

for (i = 7; i >= 0; i--) {

(x & (1 << i)) ? putchar('1') : putchar('0');

if (i % 4 == 0){

cout << " ";

}

}

printf("\n");

}

int main()

{

//cout << (getnbit2 (-9, 32)) << endl;

PrintBits(get32bit(45));

cout << endl;

int score = 0;

// 1.1

printf("1.1 bitOr");

if(bitOr(3,-9) == (3 | -9))

{

printf("\tPass.");

score += 1;

}

else

printf("\tFailed.");

//1.2

printf("\n1.2 negative");

if(negative(0) == 0 && negative(9) == -9 && negative(-5) == 5)

{

printf("\tPass.");

score += 1;

}

else

printf("\tFailed.");

//1.3

printf("\n1.3 flipBit");

if(flipBit(15,0)==14 && flipBit(0x3344,9)==0x3144 && flipBit(0,3)==8 && flipBit(0x7fffffff,31)==0xffffffff)

{

printf("\tPass.");

score += 2;

}

else

printf("\tFailed.");

//1.4

printf("\n1.4 getnbit");

if(getnbit(15,3)==7 && getnbit(63,6)==63 && getnbit(30,2)==2)

{

if(getnbit(-9,32)==-9 && getnbit(128, 33)==128)

{

printf("\nAdvanced Pass.");

score += 3;

}

else

{

printf("\tPass.");

score += 2.5;

}

}

else

printf("\tFailed.");

//1.5

printf("\n1.5 divpw2");

if(divpw2(0xfffffff,-4)==0xfffffff0 && divpw2(15,-2)== 60 && divpw2(2,-4)==32)

{

if (divpw2(10,1) == 5 && divpw2(50,2)==12)

{

printf("\tAdvanced Pass.");

score += 3;

}

else

{

printf("\tPass.");

score += 2;

}

}

else

printf("\tFailed.");

//2.1

printf("\n2.1 isSameSign");

if(isSameSign(4,10)==1 && isSameSign(-5,2)==0 && isSameSign(-5,-9)==1)

{

printf("\tPass.");

score += 2;

}

else

printf("\tFailed.");

//2.2

printf("\n2.2 is16x");

if(is16x(16)==1 && is16x(3)==0 && is16x(0)==1)

{

printf("\tPass.");

score += 2;

}

else

printf("\tFailed.");

//2.3

printf("\n2.3 isPositive");

if(isPositive(16) == 1 && isPositive(0) == 0 && isPositive(-8)==0)

{

printf("\tPass.");

score += 3;

}

else

printf("\tFailed.");

//2.4

printf("\n2.4 isLess2n");

if(isLess2n(12,4) ==1 && isLess2n(8,3)==0 && isLess2n(15,2)==0)

{

printf("\tPass.");

score += 3;

}

else

printf("\tFailed.");

printf("\n------\nYour score: %.1f", (float)score/2);

}

Câu nâng cao 1.4:

Ta biết: nếu muốn lấy n bit cuối của x thì phải AND x với 1 mask có n bit cuối là 1. Tạo mask này bằng việc shift phải 32 - n bit của 0xffffffff.

Có: 32 – n = (n^31) + 1

Code thực thi:

x & (0xffffffff >> ( (n ^ 31) + 1))

Vấn đề của logic này ở phần nâng cao là n không âm, có thể lớn hơn 31. Khi đó (n XOR 31) + 1 sẽ không còn là 32-n. Mong muốn đặt ra là nếu n > 31 thì (n^31) + 1 = 0, để AND x với 32 bit 1, còn nếu n < 31 thì (n XOR 31) + 1 = 32 – n. Điều này tương đương tồn tại 1 phép toán ? và 1 hàm f(y) sao cho:

[(n XOR 31) + 1] ? f(y) giữ nguyên (= 32-n), nếu n <= 31

hoặc

[(n XOR 31) + 1] ? f(y) = 0, nếu n > 31

Qua nhiều lần thử y, f và ?, đã tới lượt em chọn y = (n <= 31)? 1: 0 (Ta có 1 hàm như thế ở 2.4: isLess2n(2,5))

Xét việc lấy 5 bit cuối của 1 số, ta AND số đó với 0b11111. Áp dụng điều này khi n<=31, tức:

y = 1: f(1) = 0b11111

còn khi n > 31:

y = 0: f(0) = 0

Vậy chẳng phải khi y = 1, ta sẽ shift right 0b1111100000 5 lần để được 0b11111, và nếu y = 0 thì shift right 5 lần vẫn là 0b1111100000. Và khi lấy 5 bit cuối sẽ được 0.

Em phát hiện 1 hàm f(y) thỏa điều trên: f(y) = Get5bit(0b1111100000>>y>>y>>y>>y>>y)

Code:

int get5bit (int x){

int kq = x &0b11111;

return kq;

}

int getnbit(int x, int n){

int check = isLess2n (n, 5);

int kq = x & (0xffffffff >> ( ((n ^ 31) + 1) & get5bit (0b1111100000>>check>>check>>check>>check>>check) ) );

return kq;

}