**Assignment Number**

**Problem Statement**

Program in C to find the least common multiple and highest common factor of a set of integers.

**Theory**

The **highest common factor** of two or more integers, which are not all zero, is the largest positive integer that divides each of the integers. The **lowest common multiple** of two integers *a* and *b*, is the smallest positive integer that is divisible by both *a* and *b*. Since division of integers by zero is undefined, this definition has meaning only if *a* and *b* are both different from zero.

**Example :** The HCF of 25 and 20 is 5. The LCM of 25 and 20 is 100.

**Algorithm**

**Algorithm\_HCF(m, n)**

**Input :** The numbers to find HCF of, say m and n.

**Output :** The HCF of the given numbers.

**Steps :**

1. If(m == 0)

Then

* 1. Return n

1. Else If(n == 0)

Then

* 1. Return m

1. Else
   1. Return Hcf(n, m%n)

[End of if structure]

**Algorithm\_LCM(m, n, x)**

**Input :**

1. m : First(larger) number
2. n : Second(smaller) number
3. x : Increment factor

**Output :** The LCM of m and n.

**Steps :**

1. If(m % n != 0)

Then

* 1. Return Lcm(m + x, n, x)

1. Else
   1. Return m

[End of if structure]

**Algorithm\_Main()**

**Input :** A array of numbers to find the LCM and HCF of, say A.

**Output :** The LCM and HCF of the given numbers.

**Steps :**

1. Print "Enter number of elements : "
2. Input num
3. If(num < 1)

Then

* 1. Print "[Error] Number of elements must be positive!"
  2. End

[End of if structure]

1. Print "Enter 1st number : "
2. Input n
3. Set A[1] = n, lc = A[1], hc = A[1]
4. Set i = 2
5. Repeat through step 8.a to 8.m while (i <= num)
   1. Set suf = i%10
   2. Print "Enter ",i
   3. If(suf == 1)

Then

* + 1. Print "st"
  1. Else If(suf == 2)

Then

* + 1. Print "nd"
  1. Else If(suf == 3)

Then

* + 1. Print "rd"
  1. Else
     1. Print "th"

[End of if structure]

* 1. Print " number : "
  2. Input n
  3. Set A[i] = n
  4. If(lc < A[i])

Then

* + 1. Set lc = LCM(A[i], lc, A[i])
  1. Else
     1. Set lc = LCM(lc, A[i], lc)

[End of if structure]

* 1. Set hc = HCF(hc, A[i])
  2. Set i = i + 1

[End of while loop]

1. Print "\nHighest Common Factor : ", hc
2. Print "\nLowest Common Mulitple : ", lc, "\n"

**Source Code**

#include <stdio.h>

// Procedure to find HCF between m and n

int HCF(int m, int n) {

if (!m) // m is zero, so n is hcf

return n;

if (!n) // n is zero, so m is hcf

return m;

// both are non-zero, so re-divide m,

// and make it new n

// new m is old n

return (HCF(n, m % n)); // Recursive call

}

/\*

\* Procedure to find LCM between m and n

\* n : number2

\* x : number1

\* m : present i\*x, i = 1,2,3,...

\* initially n < x

\*/

int LCM(int m, int n, int x) {

if (m % n) // Remainder is greater than zero, so we're

// gonna go to the next factor of m, i.e. (m+x)

// and check if that is divisible by n

return (LCM((m + x), n, x)); // Recursive call

else

return m; // m is completely divisible by n

}

// Driver

int main() {

int n, i, lc, hc;

printf("\nEnter number of elements : ");

scanf("%d", &n);

if (n < 1) {

printf("\n[Error] Number of elements must be positive!");

return 1;

}

int A[n];

printf("Enter 1st number : ");

scanf("%d", &A[0]);

lc = hc = A[0];

for (i = 1; i < n; i++) {

int j = i + 1, suf = j % 10; // for display purposes

printf("Enter %d%s number : ", j,

suf == 1 ? "st" : // \*1st

suf == 2 ? "nd" : // \*2nd

suf == 3 ? "rd" : "th"); // \*3rd / \*th

scanf("%d", &A[i]); // input

if (lc < A[i])

lc = LCM(A[i], lc, A[i]); // LCM Function calling

else

lc = LCM(lc, A[i], lc);

hc = HCF(hc, A[i]); // HCF function calling

}

printf("\nHighest Common Factor : %d", hc);

printf("\nLowest Common Multiple : %d", lc);

}

**Input and Output**

**Set 1 :**

Enter number of elements : 2

Enter 1st number : 12

Enter 2nd number : 24

Highest Common Factor : 12

Lowest Common Multiple : 24

**Set 2 :**

Enter number of elements : 4

Enter 1st number : 5

Enter 2nd number : 12

Enter 3rd number : 20

Enter 4th number : 30

Highest Common Factor : 1

Lowest Common Multiple : 60

**Set 3 :**

Enter number of elements : 5

Enter 1st number : 15

Enter 2nd number : 30

Enter 3rd number : 27

Enter 4th number : 45

Enter 5th number : 51

Highest Common Factor : 3

Lowest Common Multiple : 4590

**Discussion**

1. For large datasets, this program is infeasible.
2. The recursive call can result to a stack overflow depending on the size of the call stack, and hence it is machine dependent.
3. For large numbers, complexity of this algorithm is high.