**Date-**

**Assignment No. :**

**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 12 and 16 is 4. The LCM of 12 and 16 is 48.

**Algorithm:**

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

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

**Steps:**

Algorithm for method HCF(m, n): //The numbers to find HCF of, say m

// and n

1. If(m == 0) Then
2. Return n
3. Else If(n == 0) Then
4. Return m
5. Else
6. Return Hcf(n, m%n)

[End of if structure]

Algorithm for method LCM(m, n, x)://m is the First(larger) number, n is

//the Second(smaller) number and x is //Increment factor

1. If(m % n != 0) Then
2. Return Lcm(m + x, n, x)
3. Else
4. Return m

[End of if structure]

Algorithm for method Main():

1. Print "Enter number of elements : "
2. Input num
3. If(num < 1) Then
4. Print "[Error] Number of elements must be positive!"

[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 10 to 28 While (i <= num)
6. Set suf = i%10
7. Print "Enter ",i
8. If(suf == 1) Then
9. Print "st"
10. Else If(suf == 2) Then
11. Print "nd"
12. Else If(suf == 3) Then
13. Print "rd"
14. Else
15. Print "th"

[End of if structure]

1. Print " number : "
2. Input n
3. Set A[i] = n
4. If(lc < A[i]) Then
5. Set lc = LCM(A[i], lc, A[i])
6. Else
7. 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;

// When both are non-zero

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

}

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

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

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

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

else

return m; // m is completely divisible by n

}

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

}

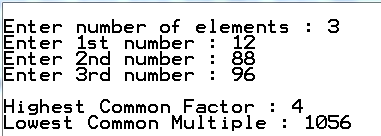
printf("\nHighest Common Factor : %d", hc);

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

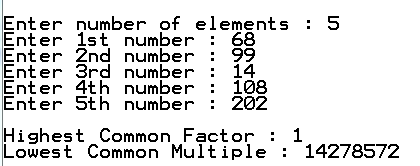
}

**Input & Output:**

Set 1:



Set 2:



**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.