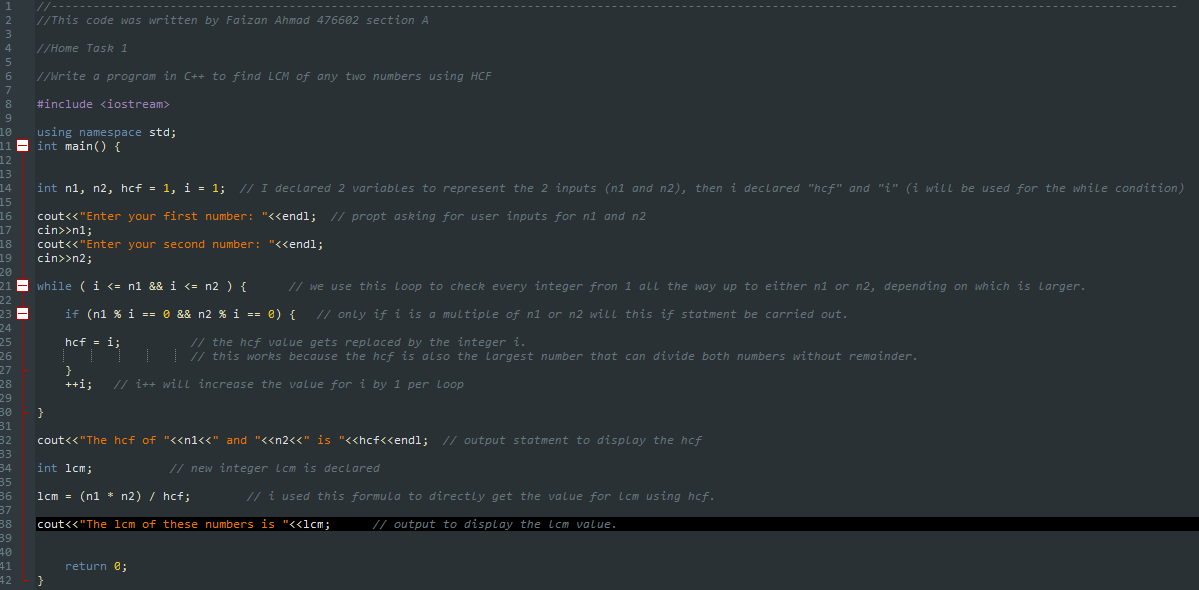
***Lab Manual 5, Home Task 1-4***

**This document was written by:**

Faizan Ahmad (476602)

Section A

ME-15

**Task 1: **

**Code:**

#include <iostream>

using namespace std;

int main() {

int n1, n2, hcf = 1, i = 1; // I declared 2 variables to represent the 2 inputs (n1 and n2), then i declared "hcf" and "i" (i will be used for the while condition)

cout<<"Enter your first number: "<<endl; // propt asking for user inputs for n1 and n2

cin>>n1;

cout<<"Enter your second number: "<<endl;

cin>>n2;

while ( i <= n1 && i <= n2 ) { // we use this loop to check every integer fron 1 all the way up to either n1 or n2, depending on which is larger.

if (n1 % i == 0 && n2 % i == 0) { // only if i is a multiple of n1 or n2 will this if statment be carried out.

hcf = i; // the hcf value gets replaced by the integer i.

// this works because the hcf is also the largest number that can divide both numbers without remainder.

}

++i; // i++ will increase the value for i by 1 per loop

}

cout<<"The hcf of "<<n1<<" and "<<n2<<" is "<<hcf<<endl; // output statment to display the hcf

int lcm; // new integer lcm is declared

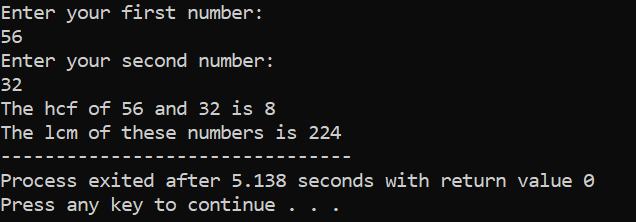
lcm = (n1 \* n2) / hcf; // i used this formula to directly get the value for lcm using hcf.

cout<<"The lcm of these numbers is "<<lcm; // output to display the lcm value.

return 0;

}

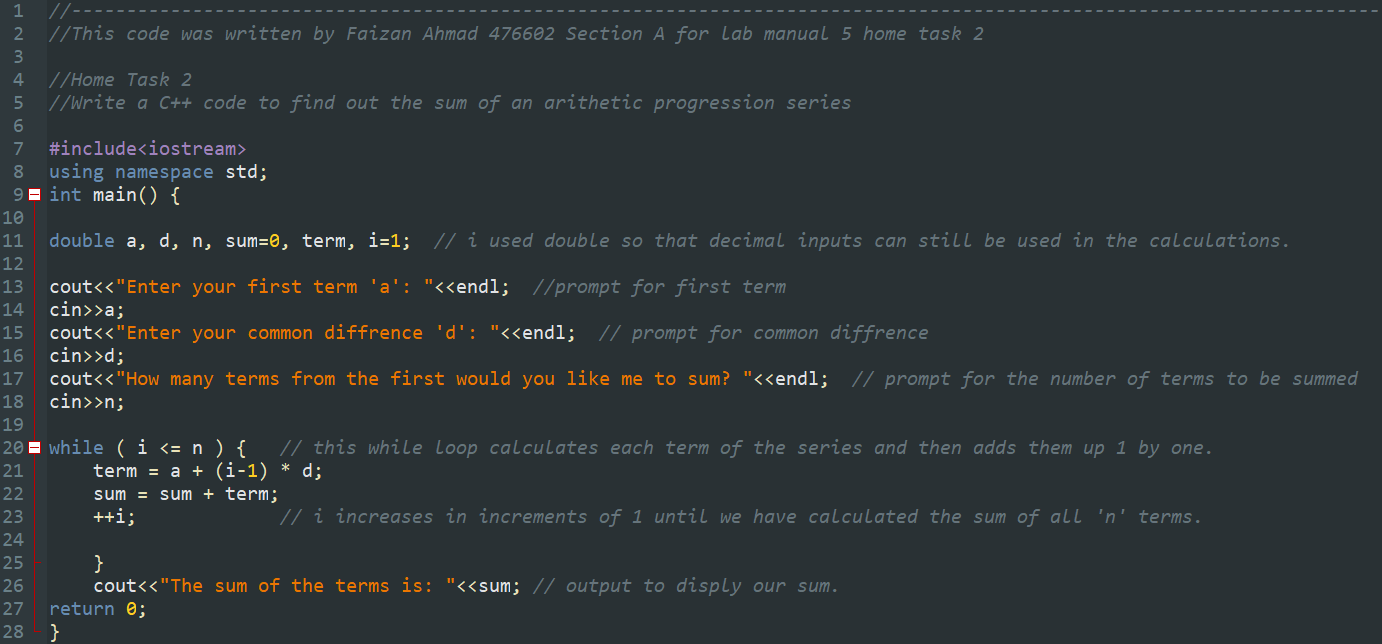
**Execute:**

****

**Explanation:**

This task required us to write a code that could find the Highest Common Factor (hcf) and Lowest Common Multiple (lcm). I started out by declaring hcf, n1, n2, lcm, and I. after taking n1 and n2 inputs the code runs a while loop that checks all values of I from 1 to n1 or n2 (depends on which one is smaller), if they can divide n1 and n2 without any remainder. The value of I is then increased by 1 after each loop and finally once the largest number that divides both n1 and n2 is determined the loop stops and the number is displayed as the hcf. To find lcm we use the formula, lcm = (n1 \* n2) / hcf. the output of this equation, lcm, is then displayed.

Task 2:



**Code:**

#include<iostream>

using namespace std;

int main() {

double a, d, n, sum=0, term, i=1; // i used double so that decimal inputs can still be used in the calculations.

cout<<"Enter your first term 'a': "<<endl; //prompt for first term

cin>>a;

cout<<"Enter your common diffrence 'd': "<<endl; // prompt for common diffrence

cin>>d;

cout<<"How many terms from the first would you like me to sum? "<<endl; // prompt for the number of terms to be summed

cin>>n;

while ( i <= n ) { // this while loop calculates each term of the series and then adds them up 1 by one.

term = a + (i-1) \* d;

sum = sum + term;

++i; // i increases in increments of 1 until we have calculated the sum of all 'n' terms.

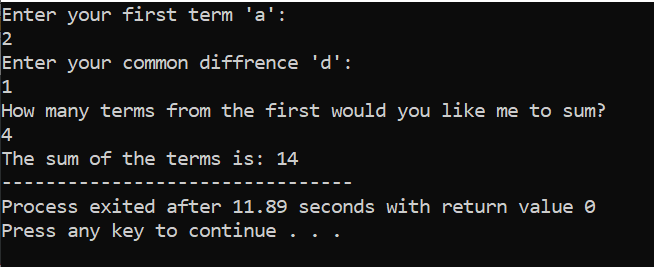
}

cout<<"The sum of the terms is: "<<sum; // output to disply our sum.

return 0;

}

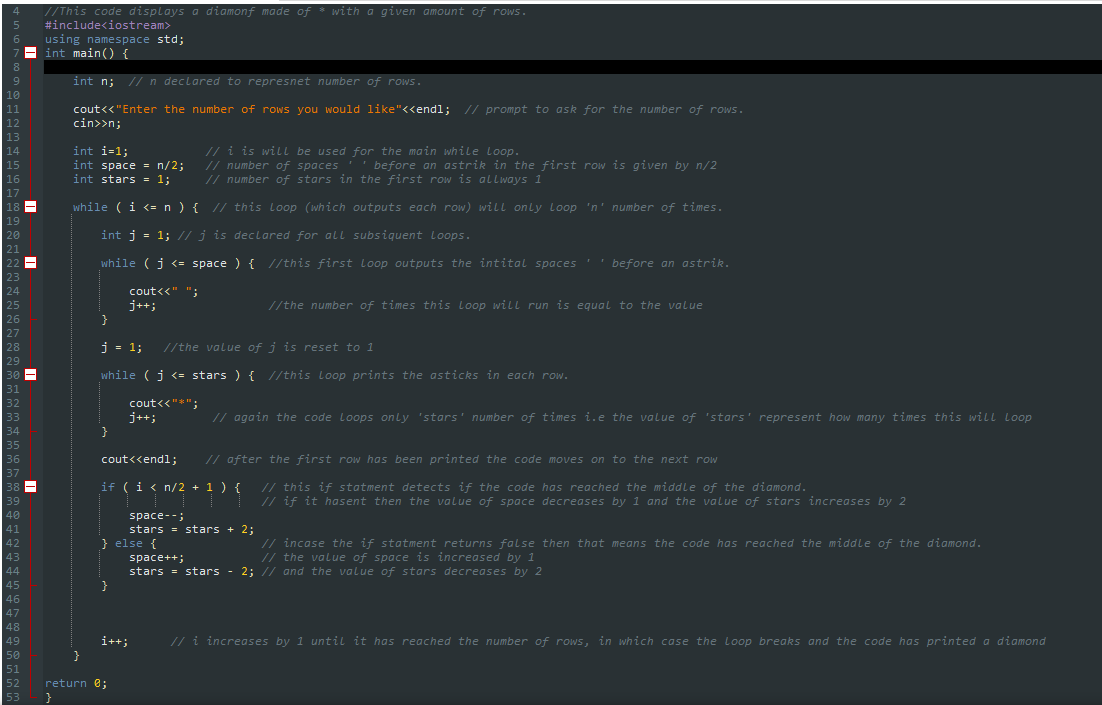
**Execute:**

****

**Explanation:**

This task required us to find the sum of first ‘n’ terms in an arithmetic series, to start of I declared ‘a’ as first term, ‘d’ as common difference, ‘term’ for each term in the series, ‘sum’ for total sum of all n terms, and ‘n’ for number of terms. Using a while loop the code calculates each term of the arithmetic series using the formula; term = a + (i-1) \* d. the term is then summed into the ‘sum’ variable and the loop breaks once ‘n’ number of terms have been calculated and summed. The final answer is then displayed.

**Task 3:**

****

**Code:**

#include<iostream>

using namespace std;

int main() {

int n; // n declared to represnet number of rows.

cout<<"Enter the number of rows you would like"<<endl; // prompt to ask for the number of rows.

cin>>n;

int i=1; // i is will be used for the main while loop.

int space = n/2; // number of spaces ' ' before an astrik in the first row is given by n/2

int stars = 1; // number of stars in the first row is allways 1

while ( i <= n ) { // this loop (which outputs each row) will only loop 'n' number of times.

int j = 1; // j is declared for all subsiquent loops.

while ( j <= space ) { //this first loop outputs the intital spaces ' ' before an astrik.

cout<<" ";

j++; //the number of times this loop will run is equal to the value

}

j = 1; //the value of j is reset to 1

while ( j <= stars ) { //this loop prints the asticks in each row.

cout<<"\*";

j++; // again the code loops only 'stars' number of times i.e the value of 'stars' represent how many times this will loop

}

cout<<endl; // after the first row has been printed the code moves on to the next row

if ( i < n/2 + 1 ) { // this if statment detects if the code has reached the middle of the diamond.

// if it hasent then the value of space decreases by 1 and the value of stars increases by 2

space--;

stars = stars + 2;

} else { // incase the if statment returns false then that means the code has reached the middle of the diamond.

space++; // the value of space is increased by 1

stars = stars - 2; // and the value of stars decreases by 2

}

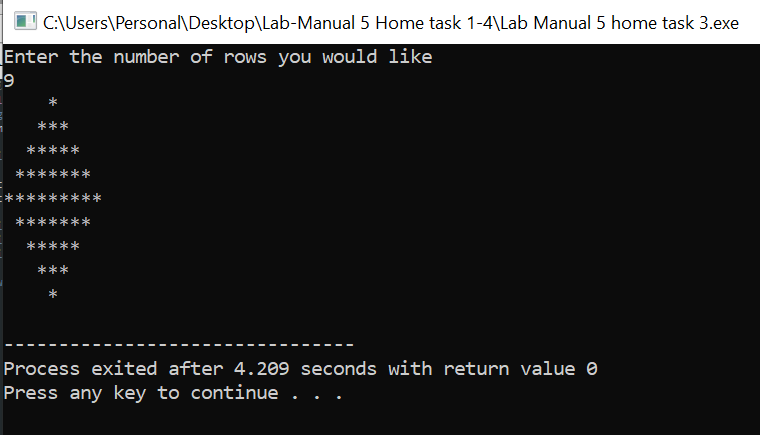
i++; // i increases by 1 until it has reached the number of rows, in which case the loop breaks and the code has printed a diamond

}

return 0;

}

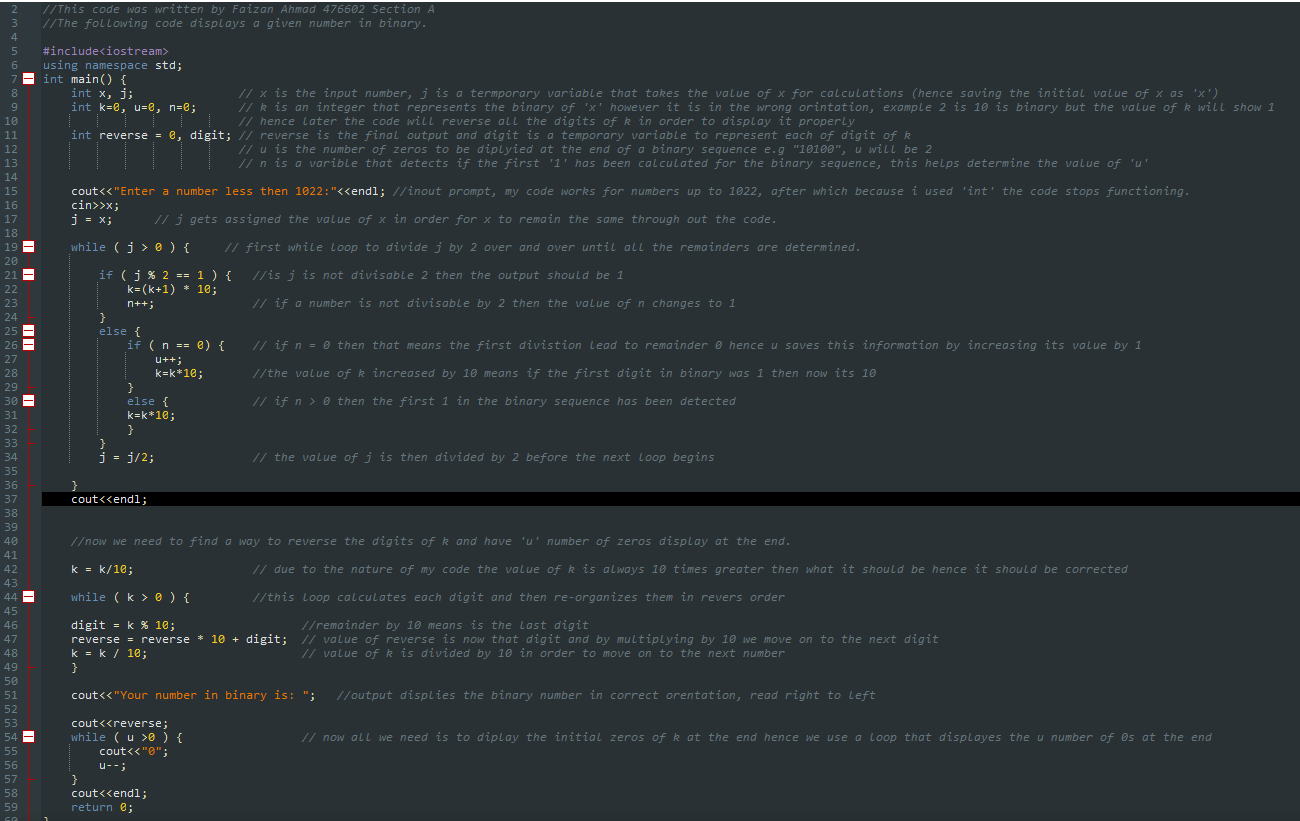
**Execute:**

****

**Explanation:**

For this task we had to make a code that prints a diamond made of asterisk (\*). My code does this by first taking the user input for number of rows, then it calculates the number of spaces in the first row, number space’s = row/2, and the number of asterisks in row 1 is always = 1. The first loop starts which loops after each row gets printed ( the loop has the condition that is will loop until I = row, I is initially 1 ). In the loop we place 2 more loops, the first prints the initial spaces in each row and the latter prints all the asterisks in the same row. Just before the main loop ends an if operation runs which checks if the code has printed the first half of the diamond, if the code has not then the value of ‘spaces’ decreases by 1 and the value of ‘stars’ increases by 2, else the value of ‘spaces’ increases by 1 and the value of ‘stars’ decreases by 2. After this the value of I increases by 1, hence when I = row the main loop ends and the code has printed a perfect diamond.

**Task 4:**

****

**Code:**

#include<iostream>

using namespace std;

int main() {

int x, j; // x is the input number, j is a termporary variable that takes the value of x for calculations (hence saving the initial value of x as 'x')

int k=0, u=0, n=0; // k is an integer that represents the binary of 'x' however it is in the wrong orintation, example 2 is 10 is binary but the value of k will show 1

// hence later the code will reverse all the digits of k in order to display it properly

int reverse = 0, digit; // reverse is the final output and digit is a temporary variable to represent each of digit of k

// u is the number of zeros to be diplyied at the end of a binary sequence e.g "10100", u will be 2

// n is a varible that detects if the first '1' has been calculated for the binary sequence, this helps determine the value of 'u'

cout<<"Enter a number less then 1022:"<<endl; //inout prompt, my code works for numbers up to 1022, after which because i used 'int' the code stops functioning.

cin>>x;

j = x; // j gets assigned the value of x in order for x to remain the same through out the code.

while ( j > 0 ) { // first while loop to divide j by 2 over and over until all the remainders are determined.

if ( j % 2 == 1 ) { //is j is not divisable 2 then the output should be 1

k=(k+1) \* 10;

n++; // if a number is not divisable by 2 then the value of n changes to 1

}

else {

if ( n == 0) { // if n = 0 then that means the first divistion lead to remainder 0 hence u saves this information by increasing its value by 1

u++;

k=k\*10; //the value of k increased by 10 means if the first digit in binary was 1 then now its 10

}

else { // if n > 0 then the first 1 in the binary sequence has been detected

k=k\*10;

}

}

j = j/2; // the value of j is then divided by 2 before the next loop begins

}

cout<<endl;

//now we need to find a way to reverse the digits of k and have 'u' number of zeros display at the end.

k = k/10; // due to the nature of my code the value of k is always 10 times greater then what it should be hence it should be corrected

while ( k > 0 ) { //this loop calculates each digit and then re-organizes them in revers order

digit = k % 10; //remainder by 10 means is the last digit

reverse = reverse \* 10 + digit; // value of reverse is now that digit and by multiplying by 10 we move on to the next digit

k = k / 10; // value of k is divided by 10 in order to move on to the next number

}

cout<<"Your number in binary is: "; //output displies the binary number in correct orentation, read right to left

cout<<reverse;

while ( u >0 ) { // now all we need is to diplay the initial zeros of k at the end hence we use a loop that displayes the u number of 0s at the end

cout<<"0";

u--;

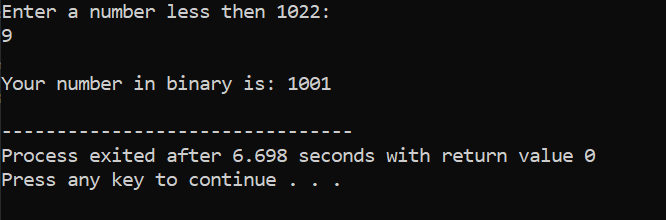
}

cout<<endl;

return 0;

}

**Execute:**



**Explanation:**

This code outputs the binary value for an input value, this is done by first checking if the number is divisible by 2, if It is then we save the first zero as a value of u. if not then the value of k is 1, the next loop begins after the number is divided by 2, the new value Is again tested and over time the value of k changes to the binary of the original value. However, the initial zeros cannot be saved e.g, binary of 6 is 110 but value of k is 11 and u is 1. Hence, we must rearrange the values of k and place the ‘u’ number of zeros at the end. Hence, we use a loop to detect the digits of k and then place the new value in the integer ‘reverse’. Now the ‘reverse’ is outputted and then the value of u is outputted at the end so using the above example, output would be 110.