#include <iostream>

using namespace std;

int greatestcommon(int x, int y)

{

    if ((x % y) == 0)

        return y;

    else

        greatestcommon(y, x % y);

}

int main()

{

    int m = 0, n = 0;

    cout << "Enter an integer: ";

    cin >> m;

    cout << "Enter a smaller integer: ";

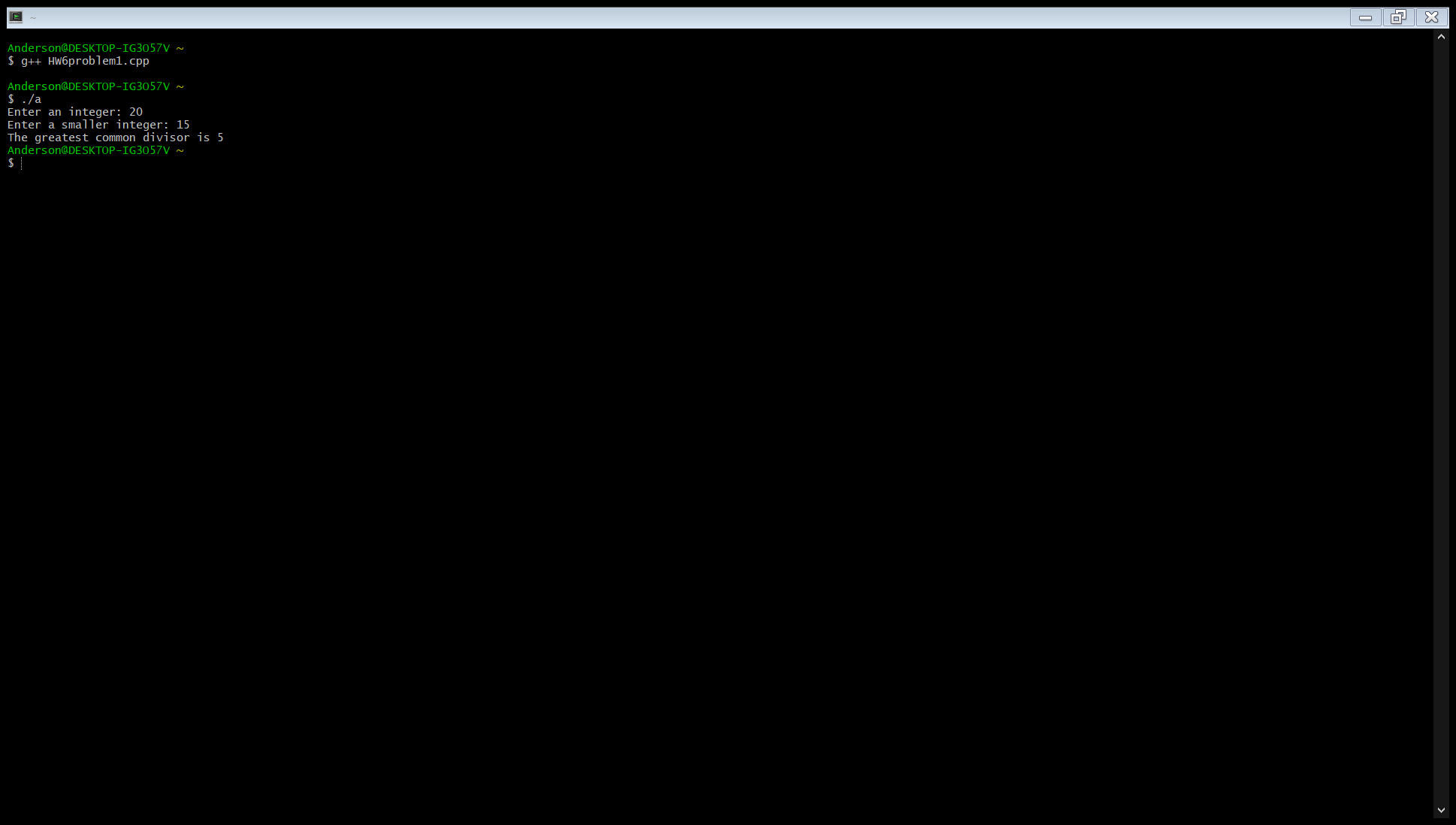
    cin >> n;

    cout << "The greatest common divisor is ";

    cout << greatestcommon(m, n);

    return 0;

}



#include <iostream>

using namespace std;

void reverseDisplay(int value)

{

    if (value > 0)

    {

        cout << value % 10;

        reverseDisplay(value/10);

    }

}

int main()

{

    int number = 0;

    cout << "Enter an integer: ";

    cin >> number;

    cout << "Your number in reverse is ";

    reverseDisplay(number);

    return 0;

}



#include<iostream>

using namespace std;

int i = 0;

string decimalToHex(int value)

{

    char characterArray[100];

    if (value!=0)

    {

        int temp = 0;

        temp = value % 16;

        if(temp <= 10)

        {

            characterArray[i] = temp + 48;

            i++;

        }

        else

        {

            characterArray[i] = temp + 55;

            i++;

        }

        temp = value/16;

        decimalToHex(temp);

    }

    string hex;

    for(int j=i-1; j>=0; j--)

        hex += characterArray[j];

    return hex;

}

// Driver program to test above function

int main()

{

    int n;

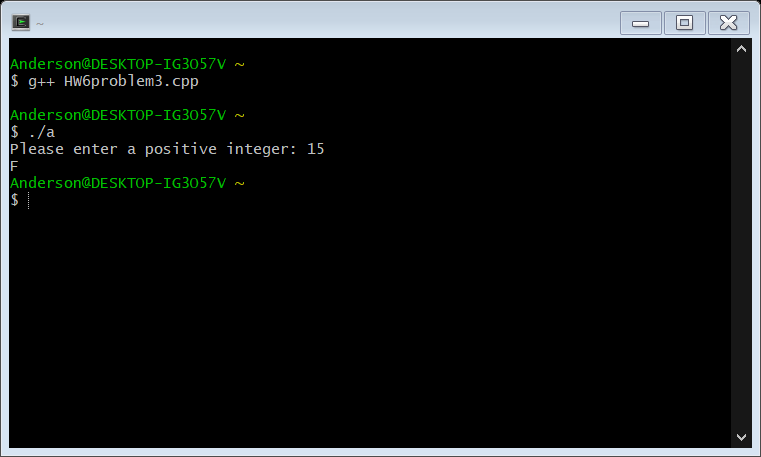
    cout << "Please enter a positive integer: ";

    cin >> n;

    cout << decimalToHex(n);

    return 0;

}



#include<iostream>

using namespace std;

const int N = 8;

void printBoard(int board[N][N])

{

   for (int i = 0; i < N; i++)

   {

      for (int j = 0; j < N; j++)

         cout << board[i][j] << " ";

         cout << endl;

   }

}

bool isValid(int board[N][N], int row, int column)

{

   for (int i = 0; i < column; i++)

      if (board[row][i])

         return false;

   for (int i = row, j = column; i >= 0 && j >= 0; i--, j--)

      if (board[i][j])

         return false;

   for (int i = row, j = column; j >= 0 && i < N; i++, j--)

      if (board[i][j])

         return false;

   return true;

}

bool solveNQueen(int board[N][N], int column)

{

   if (column >= N)

      return true;

   for (int i = 0; i < N; i++)

   {

      if (isValid(board, i, column) )

      {

         board[i][column] = 1;

         if ( solveNQueen(board, column + 1))

            return true;

         board[i][column] = 0;

      }

   }

   return false;

}

bool checkSolution() {

   int board[N][N];

   for(int i = 0; i<N; i++)

   for(int j = 0; j<N; j++)

   board[i][j] = 0;

   if (solveNQueen(board, 0) == false)

   {

      cout << "No solution";

      return false;

   }

   printBoard(board);

   return true;

}

int main()

{

    checkSolution();

    return 0;

}

