PROGRAMMING FUNDAMENTAL ASSIGNMENT

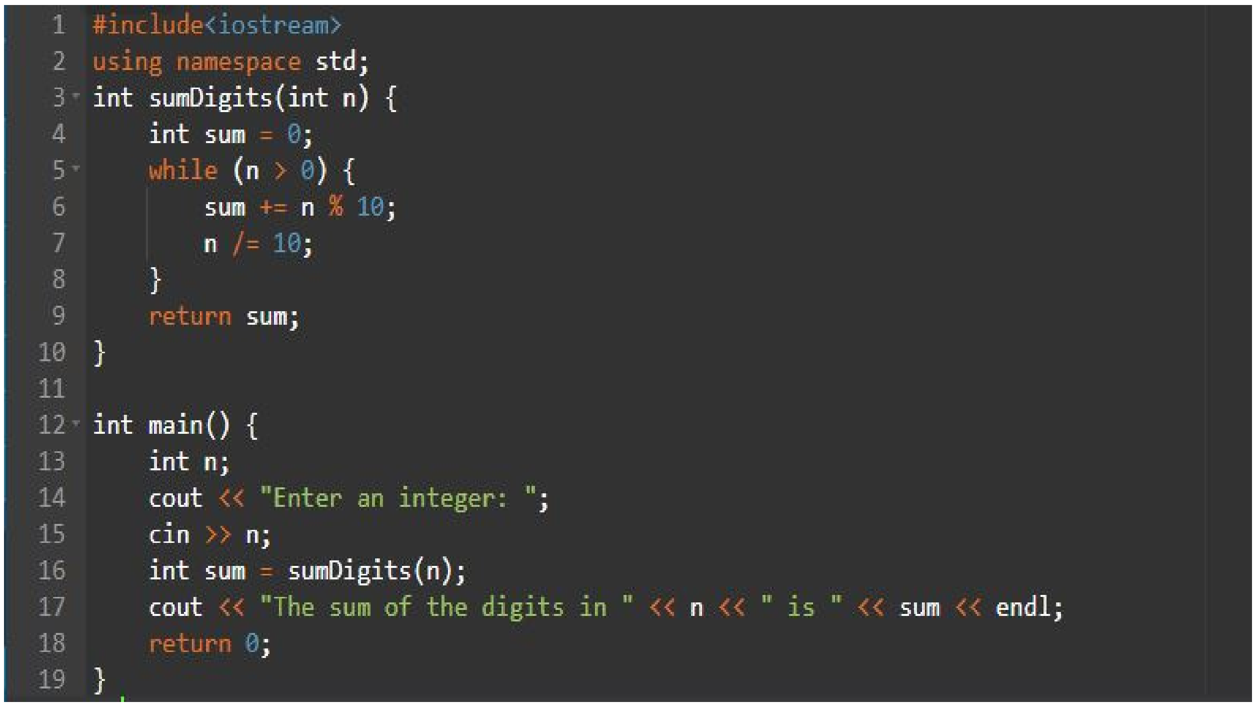
SIR ASHFAQ

ROLL NO:232481

BY: DUAA DARA

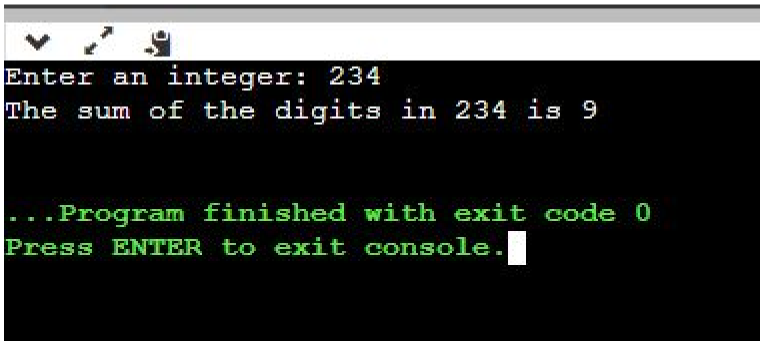
QUESTION 1:

Code:



```cpp
#include<iostream>
using namespace std;
int sumDigits(int n) {
    int sum = 0;
    while (n > 0) {
        sum += n % 10;
        n /= 10;
    }
    return sum;
}

int main() {
    int n;
    cout << "Enter an integer: ";
    cin >> n;
    int sum = sumDigits(n);
    cout << "The sum of the digits in " << n << " is " << sum << endl;
    return 0;
}
```

Output:



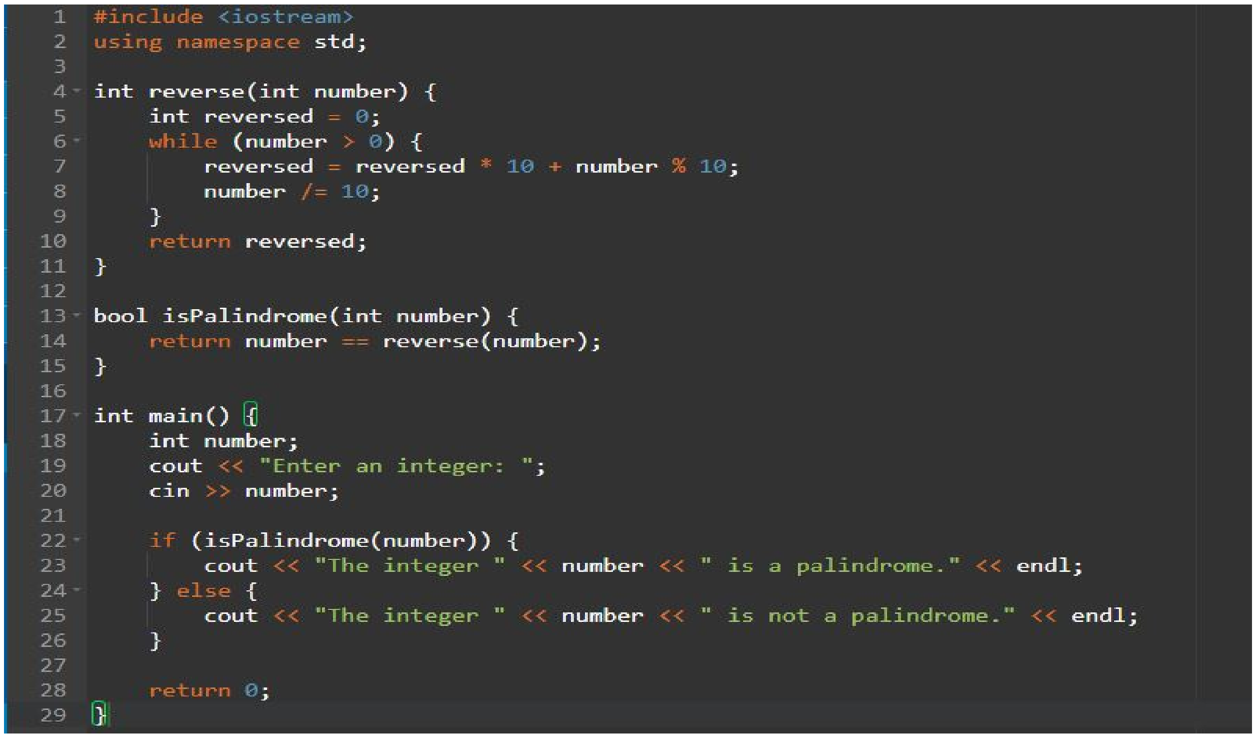
```
Enter an integer: 234
The sum of the digits in 234 is 9


...Program finished with exit code 0
Press ENTER to exit console.
```

Explanation:

This code uses a while loop to repeatedly remove the digits from the integer until all the digits are removed. The sum of the digits is calculated by adding the last digit (n % 10) to the sum.

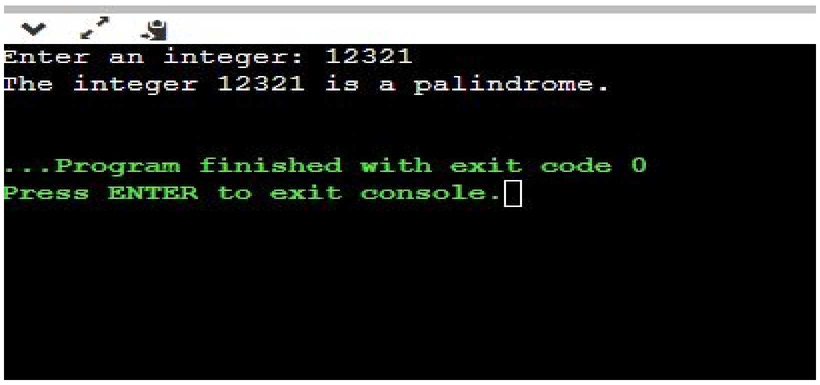
QUESTION 2:

Code:



```cpp
#include <iostream>
using namespace std;

int reverse(int number) {
    int reversed = 0;
    while (number > 0) {
        reversed = reversed * 10 + number % 10;
        number /= 10;
    }
    return reversed;
}

bool isPalindrome(int number) {
    return number == reverse(number);
}

int main() {
    int number;
    cout << "Enter an integer: ";
    cin >> number;

    if (isPalindrome(number)) {
        cout << "The integer " << number << " is a palindrome." << endl;
    } else {
        cout << "The integer " << number << " is not a palindrome." << endl;
    }

    return 0;
}
```

Output:



```
Enter an integer: 12321
The integer 12321 is a palindrome.


...Program finished with exit code 0
Press ENTER to exit console.
```

Explanation:

TWO FUNCTIONS:

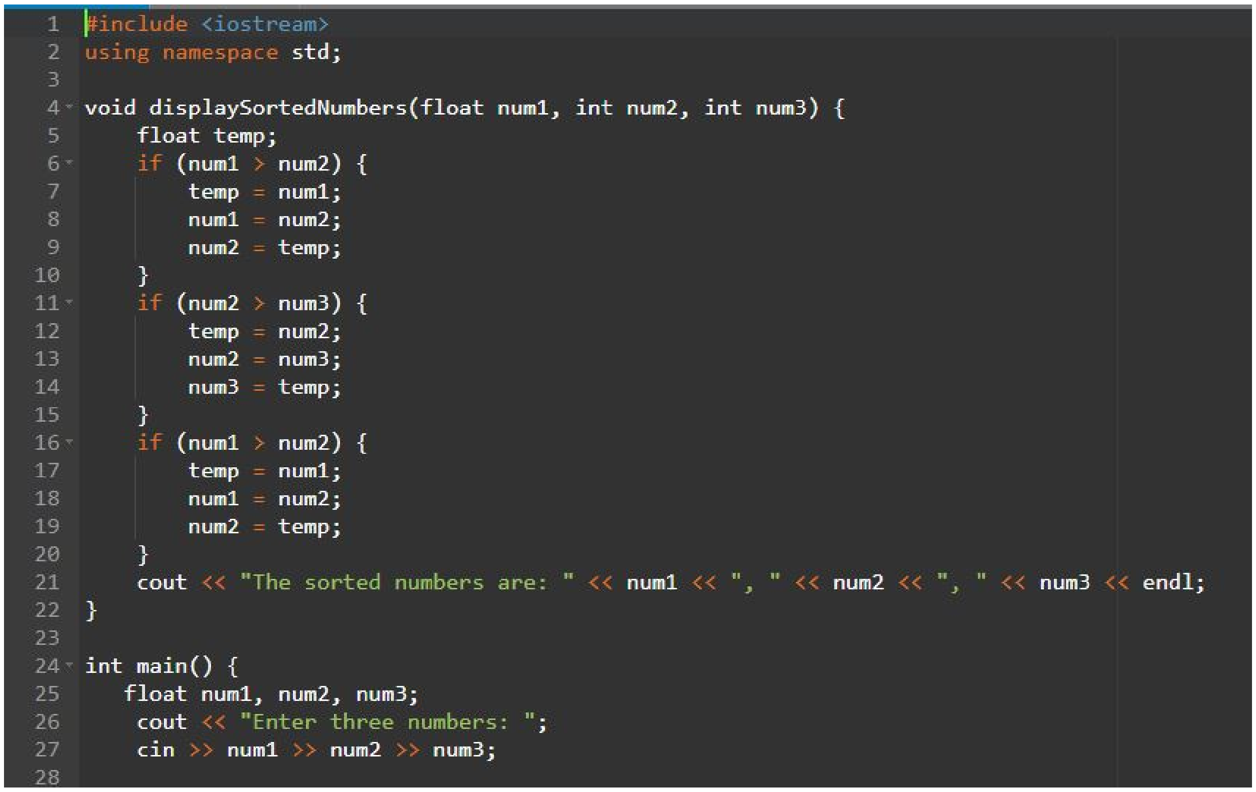
The **reverse function** reverses an integer by extracting the last digit of the integer and appending it to the reversed number.

The i**sPalindrome function** checks if a number is a palindrome by comparing the number with its reversal. A number is a palindrome if its reversal is the same as itself.

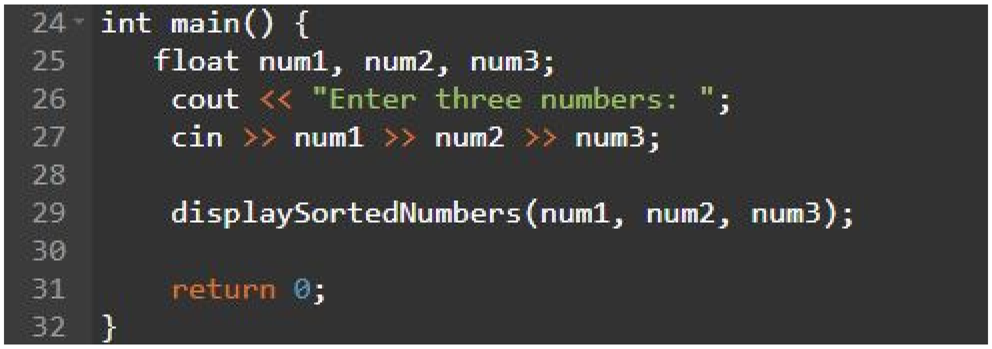
If the integer is a palindrome, the program will display a message saying "The integer X is a palindrome." If the integer is not a palindrome, the program will display a message saying "The integer X is not a palindrome."

Question 3:

Code:

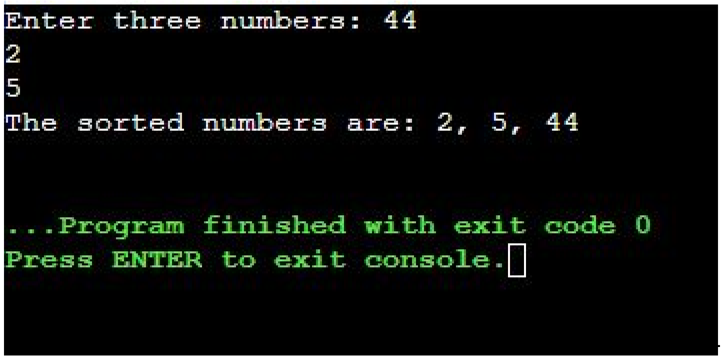


```cpp
#include <iostream>
using namespace std;

void displaySortedNumbers(float num1, int num2, int num3) {
    float temp;
    if (num1 > num2) {
        temp = num1;
        num1 = num2;
        num2 = temp;
    }
    if (num2 > num3) {
        temp = num2;
        num2 = num3;
        num3 = temp;
    }
    if (num1 > num2) {
        temp = num1;
        num1 = num2;
        num2 = temp;
    }
    cout << "The sorted numbers are: " << num1 << ", " << num2 << ", " << num3 << endl;
}

int main() {
   float num1, num2, num3;
    cout << "Enter three numbers: ";
    cin >> num1 >> num2 >> num3;

```



```cpp
int main() {
   float num1, num2, num3;
    cout << "Enter three numbers: ";
    cin >> num1 >> num2 >> num3;

    displaySortedNumbers(num1, num2, num3);

    return 0;
}
```

Output:



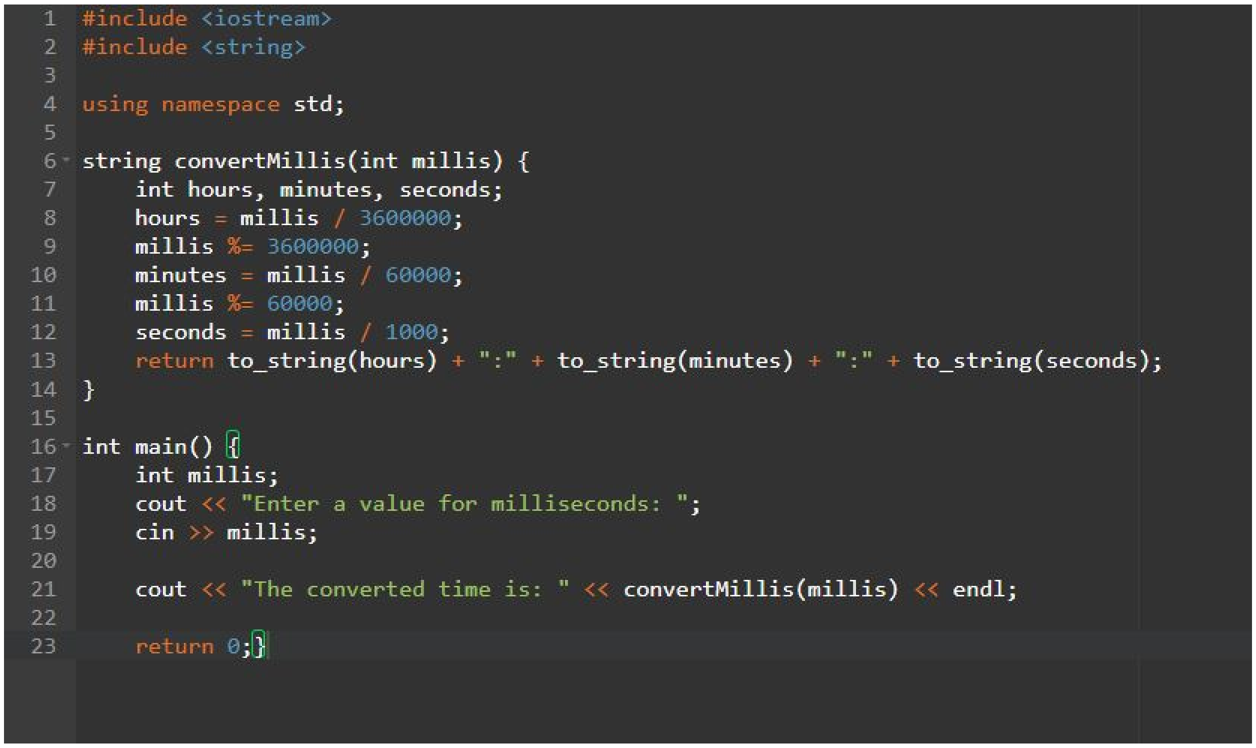
```
Enter three numbers: 44
2
5
The sorted numbers are: 2, 5, 44


...Program finished with exit code 0
Press ENTER to exit console.
```

EXPLANATION:

This program prompts the user to enter three numbers. It then uses the **display sorted numbers** function to sort the numbers in increasing order and display them. The function first compares the three numbers and swaps them if they are not in the correct order. This process is repeated until the numbers are sorted.

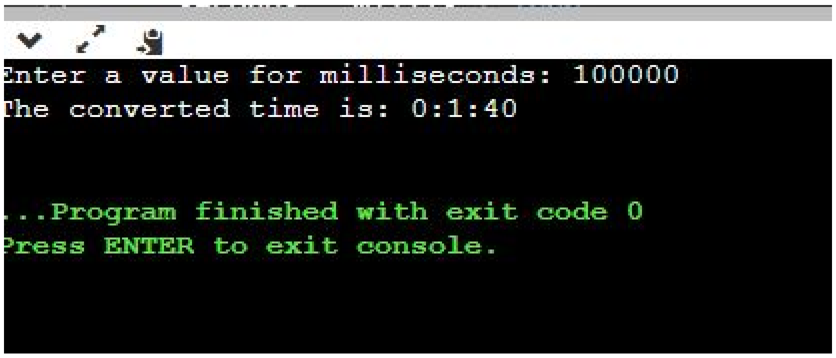
Question 4:

Code:



```cpp
#include <iostream>
#include <string>

using namespace std;

string convertMillis(int millis) {
    int hours, minutes, seconds;
    hours = millis / 3600000;
    millis %= 3600000;
    minutes = millis / 60000;
    millis %= 60000;
    seconds = millis / 1000;
    return to_string(hours) + ":" + to_string(minutes) + ":" + to_string(seconds);
}

int main() {
    int millis;
    cout << "Enter a value for milliseconds: ";
    cin >> millis;

    cout << "The converted time is: " << convertMillis(millis) << endl;

    return 0;}
```

Output:



```
Enter a value for milliseconds: 100000
The converted time is: 0:1:40


...Program finished with exit code 0
Press ENTER to exit console.
```

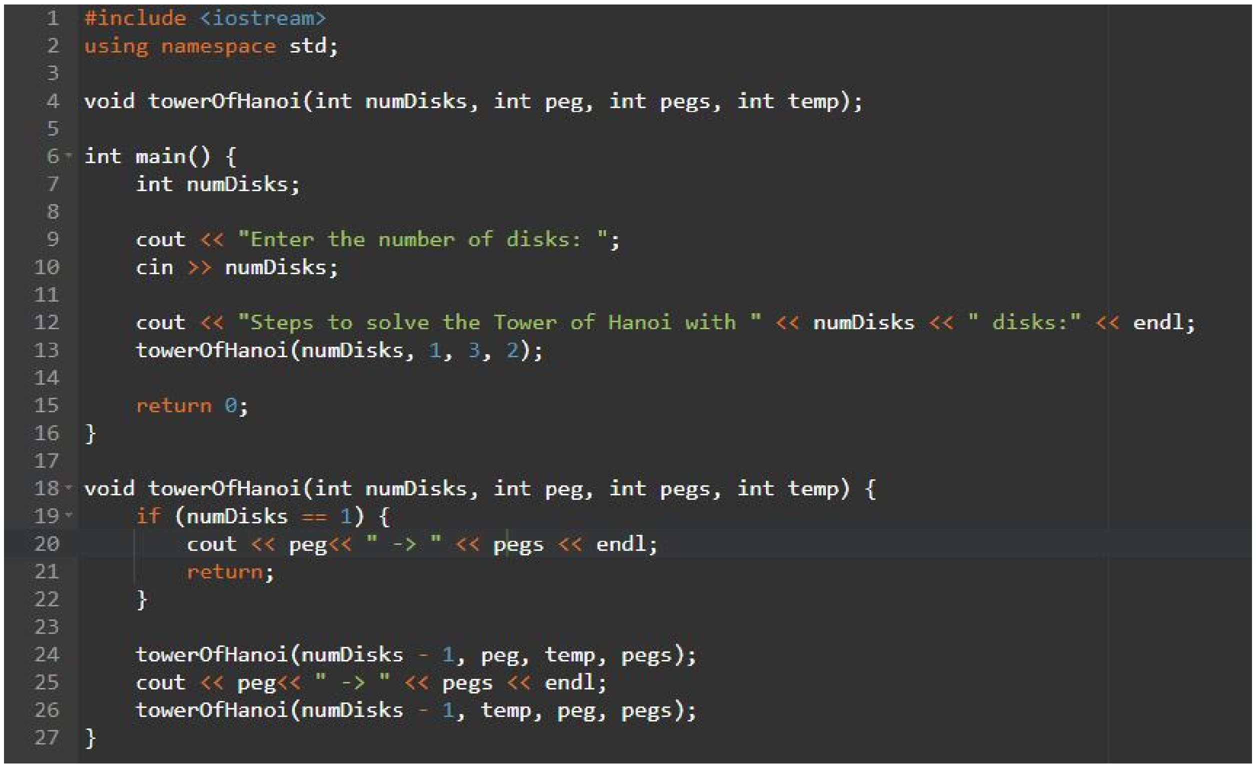
EXPLANATION:

In this program, the **convertMillis** function calculates the hours, minutes, and seconds from the given milliseconds. The function first calculates the number of hours by dividing the milliseconds by 3600000. Then, it calculates the number of minutes by dividing the remaining milliseconds by 60000. Finally, it calculates the number of seconds by dividing the remaining milliseconds by 1000. The function then returns a string in the format of hours:minutes:seconds.

Tostring function is used to convert an integer to a string. This function is a member of the std namespa

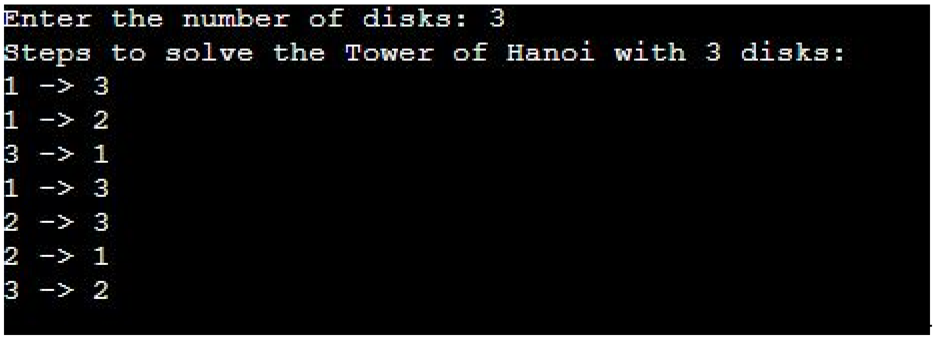
QUESTION 5:

Code:



```cpp
#include <iostream>
using namespace std;

void towerOfHanoi(int numDisks, int peg, int pegs, int temp);

int main() {
    int numDisks;

    cout << "Enter the number of disks: ";
    cin >> numDisks;

    cout << "Steps to solve the Tower of Hanoi with " << numDisks << " disks:" << endl;
    towerOfHanoi(numDisks, 1, 3, 2);

    return 0;
}

void towerOfHanoi(int numDisks, int peg, int pegs, int temp) {
    if (numDisks == 1) {
        cout << peg<< " -> " << pegs << endl;
        return;
    }

    towerOfHanoi(numDisks - 1, peg, temp, pegs);
    cout << peg<< " -> " << pegs << endl;
    towerOfHanoi(numDisks - 1, temp, peg, pegs);
}
```

Code:



```
Enter the number of disks: 3
Steps to solve the Tower of Hanoi with 3 disks:
1 -> 3
1 -> 2
3 -> 1
1 -> 3
2 -> 3
2 -> 1
3 -> 2
```

EXPLANANTION:

This C++ code is a solution to the Tower of Hanoi problem using recursion.First, the user is prompted to enter the number of disks. Then, the program calculates and prints the steps to solve the Tower of Hanoi with the specified number of disks. This recursive solution correctly prints the steps to solve the Tower of Hanoi problem with the given number of disks.