```
##################################################
##################################################
##################################################
                | Tower Of Hanoi |
##################################################
##################################################
##################################################
```

Code:

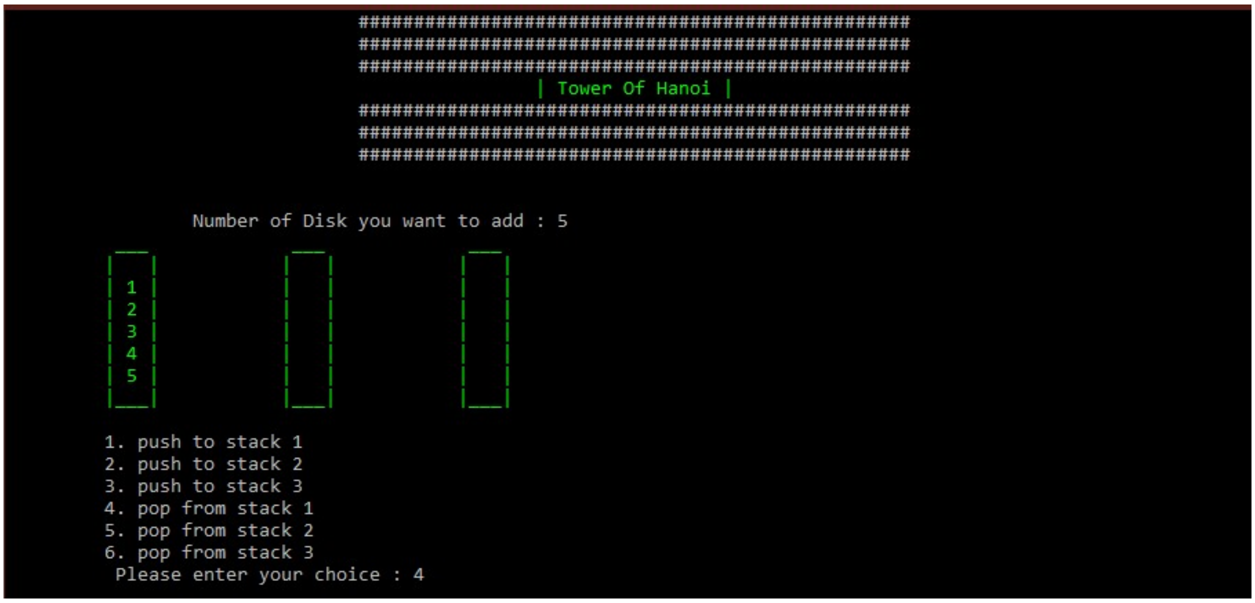
```
#include<iostream>
#include<windows.h>
using namespace std;
struct node1
{
    int data;
    node1* next;
};
struct node1* top;
int inc1;
struct node2
{
    int data;
    node2* next;
};
struct node2* top1;
int inc2;
struct node3
{
    int data;
    node3* next;
};
struct node3* top2;
int inc3;
bool push(int val, int disk)
{
    struct node1* newnode = new node1;
    newnode->data = val;
    node1* temp = top;
    if (inc1 > disk)
    {
        return false;
    }
    else
    {
        if (top == NULL)
        {
            newnode->next = top;
            top = newnode;
            inc1++;
            return true;
        }
        else
        {
            if (val >= temp->data)
```

```
                        {
                                return false;
                        }
                        else
                        {
                                newnode->next = top;
                                top = newnode;
                                inc1++;
                                return true;
                        }
                }
        }
}
bool push1(int val, int disk)
{
        struct node2* newnode = new node2;
        newnode->data = val;
        node2* temp = top1;
        if (inc2 > disk)
        {
                return false;
        }
        else
        {
                if (top1 == NULL)
                {
                        newnode->next = top1;
                        top1 = newnode;
                        inc2++;
                        return true;
                }
                else
                {
                        if (val >= temp->data)
                        {
                                return false;
                        }
                        else
                        {
                                newnode->next = top1;
                                top1 = newnode;
                                inc2++;
                                return true;
                        }
                }
        }
}
bool push2(int val, int disk)
{
        struct node3* newnode = new node3;
        newnode->data = val;
        node3* temp = top2;
        if (inc3 > disk)
        {
                return false;
        }
        else if (inc3 == 5)
        {
                cout << endl;
                cout << "\t\t |YOU WIN THE GAME |" << endl;
        }
        else
        {
                if (top2 == NULL)
                {
                        newnode->next = top2;
                        top2 = newnode;
                        inc3++;
                        return true;
                }
                else
                {
                        if (val >= temp->data)
                        {
                                return false;
                        }
                        else
                        {
                                newnode->next = top2;
                                top2 = newnode;
                                inc3++;
                                return true;
                        }
                }
        }
}
int pop()
{
        int element;
        if (top == NULL)
                cout << "Stack Underflow" << endl;
        else
        {
                element = top->data;
                top = top->next;
                inc1--;
        }
        return element;
}
int pop1()
{
        int element;
        if (top1 == NULL)
                cout << "Stack Underflow" << endl;
        else
        {
                element = top1->data;
                top1 = top1->next;
                inc2--;
        }
        return element;
}
int pop2()
{
        int element;
        if (top2 == NULL)
                cout << "Stack Underflow" << endl;
        else
        {
                element = top2->data;
                top2 = top2->next;
                inc3--;
        }
        return element;
}
void display()
{
        node1* temp = top;
        node2* temp1 = top1;
        node3* temp2 = top2;
        HANDLE colors = GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(colors, 10);
        cout << "\t  ___" << "\t\t  ___" << "\t\t  ___" << endl;
        cout << "\t |   |" << "\t\t |   |" << "\t\t |   |" << endl;
        while (temp != NULL || temp1 != NULL || temp2 != NULL)
        {
                if (temp == NULL)
                {
                        cout << "\t | " << " " << " |";
                }
                if (temp != NULL)
                {
                        cout << "\t | " << temp->data << " |";
                        temp = temp->next;
                }
                if (temp1 == NULL)
                {
                        cout << "\t\t | " << " " << " |";
                }
                if (temp1 != NULL)
                {
                        cout << "\t\t | " << temp1->data << " |";
                        temp1 = temp1->next;
                }
                if (temp2 == NULL)
                {
                        cout << "\t\t | " << " " << " |" << endl;
                }
                if (temp2 != NULL)
                {
                        cout << "\t\t | " << temp2->data << " |" << endl;
                        temp2 = temp2->next;
```

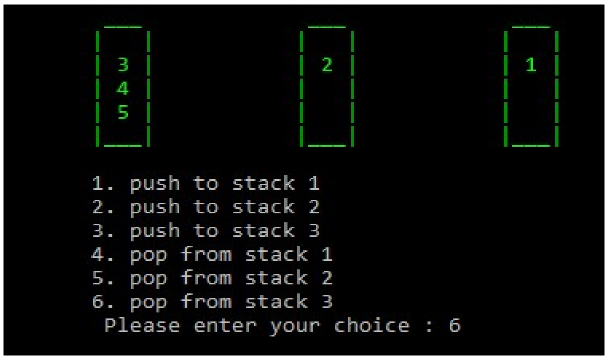
```
			}
		}
		cout << "\t |___|" << "\t\t |___|" << "\t\t |___|" << endl;
		SetConsoleTextAttribute(colors, 7);
}
int main()
{
		HANDLE colors = GetStdHandle(STD_OUTPUT_HANDLE);
		int variable = 3;
		char character = '#';
		SetConsoleTextAttribute(colors, 7);
		for (int i = 0; i < variable; i++)
		{
			cout << "\t\t\t\t";
			for (int j = 0; j < 50; j++)
			{
				//Sleep(70);
				cout << character;
			}
			cout << endl;
		}
		SetConsoleTextAttribute(colors, 10);
		//Sleep(70);
		cout << "\t\t\t\t\t	| Tower Of Hanoi | " << endl;
		SetConsoleTextAttribute(colors, 7);
		for (int i = 0; i < variable; i++)
		{
			cout << "\t\t\t\t";
			for (int j = 0; j < 50; j++)
			{
				//Sleep(70);
				cout << character;
			}
			cout << endl;
		}
		cout << endl << endl;
		int disk;
		cout << "\t\t Number of Disk you want to add : ";
		cin >> disk;
		for (int i = disk; i >= 1; i--)
		{
			push(i,disk);
		}
		int popelement1, popelement2, popelement3;
		int i;
		int elementtopushback1, elementtopushback2, elementtopushback3;
		int choice;
		do
		{
			display();
			cout << endl;
			cout << "\t 1. push to stack 1" << endl;
			cout << "\t 2. push to stack 2" << endl;
			cout << "\t 3. push to stack 3" << endl;
cout << "\t 4. pop from stack 1" << endl;
cout << "\t 5. pop from stack 2" << endl;
cout << "\t 6. pop from stack 3" << endl;
cout << "\t  Please enter your choice : ";
cin >> choice;
switch (choice)
{
case 1:
        if (i == 2)
        {
                elementtopushback2 = push(popelement2, disk);
                if (elementtopushback2 == 0)
                {
                        push1(popelement2, disk);
                }
        }
        else if (i == 3)
        {
                elementtopushback3 = push(popelement3, disk);
                if (elementtopushback3 == 0)
                {
                        push(popelement3, disk);
                }
        }
        break;
case 2:
        if (i == 1)
        {
                elementtopushback1 = push1(popelement1, disk);
                if (elementtopushback1 == 0)
                {
                        push(popelement1, disk);
                }
        }
        else if (i == 3)
        {
                elementtopushback3 = push1(popelement3, disk);
                if (elementtopushback3 == 0)
                {
                        push2(popelement3, disk);
                }
        }
        break;
case 3:
        if (i == 1)
        {
                elementtopushback1 = push2(popelement1, disk);
                if (elementtopushback1 == 0)
                {
                        push(popelement1, disk);
                }
        }
        else if (i == 2)
        {
```

```
                    elementtopushback2 = push2(popelement2, disk);
                    if (elementtopushback2 == 0)
                    {
                            push1(popelement2, disk);
                    }
                }
                break;
        case 4:
                popelement1 = pop();
                cout << "\t\t\t The disk in the hand is : " << popelement1 << endl;
                i = 1;
                break;
        case 5:
                i = 2;
                popelement2 = pop1();
                cout << "\t\t\t The disk in the hand is : " << popelement2 << endl;
                break;
        case 6:
                i = 3;
                popelement3 = pop2();
                cout << "\t\t\t The disk in the hand is : " << popelement3 << endl;
                break;
        }
        system("cls");
    } while (choice != 10);
}
```

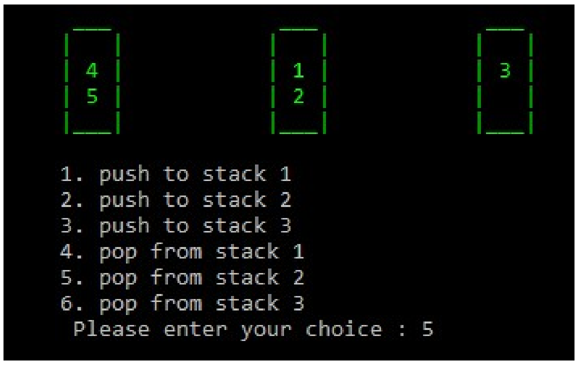
Output:



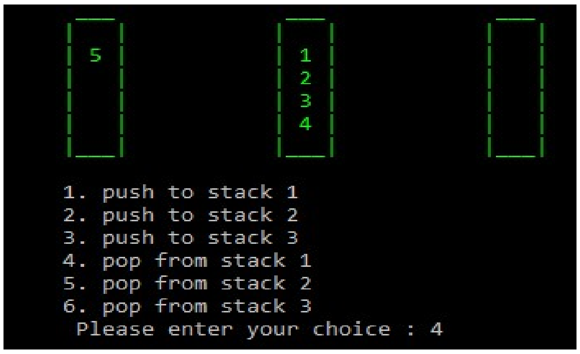
```
                    ##################################################
                    ##################################################
                    ##################################################
                                    | Tower Of Hanoi |
                    ##################################################
                    ##################################################
                    ##################################################


                 Number of Disk you want to add : 5
         ___              ___              ___
        |   |            |   |            |   |
        | 1 |            |   |            |   |
        | 2 |            |   |            |   |
        | 3 |            |   |            |   |
        | 4 |            |   |            |   |
        | 5 |            |   |            |   |
        |___|            |___|            |___|

        1. push to stack 1
        2. push to stack 2
        3. push to stack 3
        4. pop from stack 1
        5. pop from stack 2
        6. pop from stack 3
         Please enter your choice : 4
```



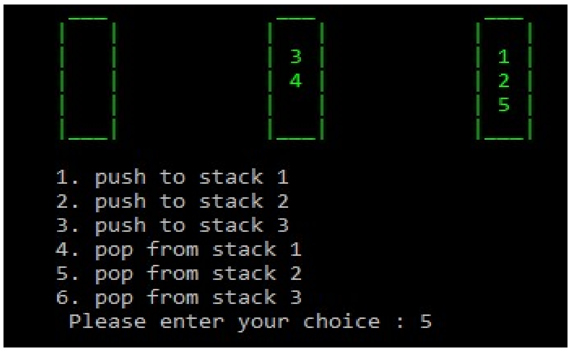
```
    ___              ___              ___
   |   |            |   |            |   |
   | 3 |            | 2 |            | 1 |
   | 4 |            |   |            |   |
   | 5 |            |   |            |   |
   |___|            |___|            |___|

   1. push to stack 1
   2. push to stack 2
   3. push to stack 3
   4. pop from stack 1
   5. pop from stack 2
   6. pop from stack 3
    Please enter your choice : 6
```



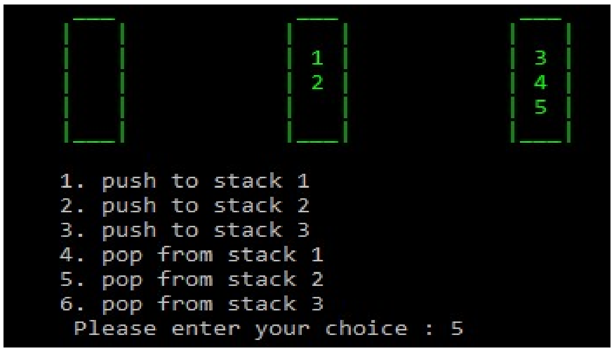
```
    ___              ___              ___
   |   |            |   |            |   |
   | 4 |            | 1 |            | 3 |
   | 5 |            | 2 |            |   |
   |___|            |___|            |___|

   1. push to stack 1
   2. push to stack 2
   3. push to stack 3
   4. pop from stack 1
   5. pop from stack 2
   6. pop from stack 3
    Please enter your choice : 5
```



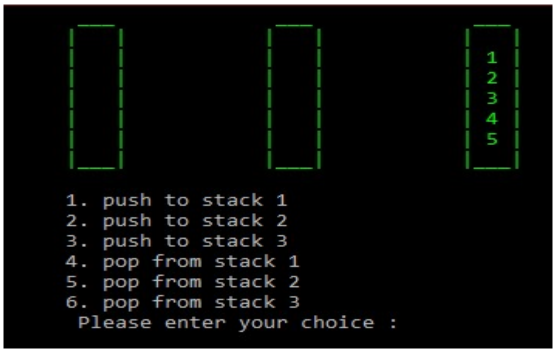
```
    ___              ___              ___
   |   |            |   |            |   |
   | 5 |            | 1 |            |   |
   |   |            | 2 |            |   |
   |   |            | 3 |            |   |
   |   |            | 4 |            |   |
   |___|            |___|            |___|

   1. push to stack 1
   2. push to stack 2
   3. push to stack 3
   4. pop from stack 1
   5. pop from stack 2
   6. pop from stack 3
    Please enter your choice : 4
```



```
    ___              ___              ___
   |   |            |   |            |   |
   |   |            | 3 |            | 1 |
   |   |            | 4 |            | 2 |
   |   |            |   |            | 5 |
   |___|            |___|            |___|

   1. push to stack 1
   2. push to stack 2
   3. push to stack 3
   4. pop from stack 1
   5. pop from stack 2
   6. pop from stack 3
    Please enter your choice : 5
```



```
    ___              ___              ___
   |   |            |   |            |   |
   |   |            | 1 |            | 3 |
   |   |            | 2 |            | 4 |
   |   |            |   |            | 5 |
   |___|            |___|            |___|

   1. push to stack 1
   2. push to stack 2
   3. push to stack 3
   4. pop from stack 1
   5. pop from stack 2
   6. pop from stack 3
    Please enter your choice : 5
```



```
    ___              ___              ___
   |   |            |   |            |   |
   |   |            |   |            | 1 |
   |   |            |   |            | 2 |
   |   |            |   |            | 3 |
   |   |            |   |            | 4 |
   |   |            |   |            | 5 |
   |___|            |___|            |___|

   1. push to stack 1
   2. push to stack 2
   3. push to stack 3
   4. pop from stack 1
   5. pop from stack 2
   6. pop from stack 3
    Please enter your choice :
```