CSIR CODING ASSIGNMENT

-SUBMITTED BY:
PREKSHA RAKHECHA
(106119095)

QUESTIONS ALLOTED:

30 b) Let Z be the set of integers {0, 1..., n-1} let (*) be the binary operator on Z such that a(*)b=the remainder of a and b divide by n

i) construct a table for binary operation (*) for n=7
ii) show that (Z, (*)) is a semigroup for any n

Construct a table for binary operation (*) for n=7

LANGUAGE USED: C++

CODE EXPALNATION:

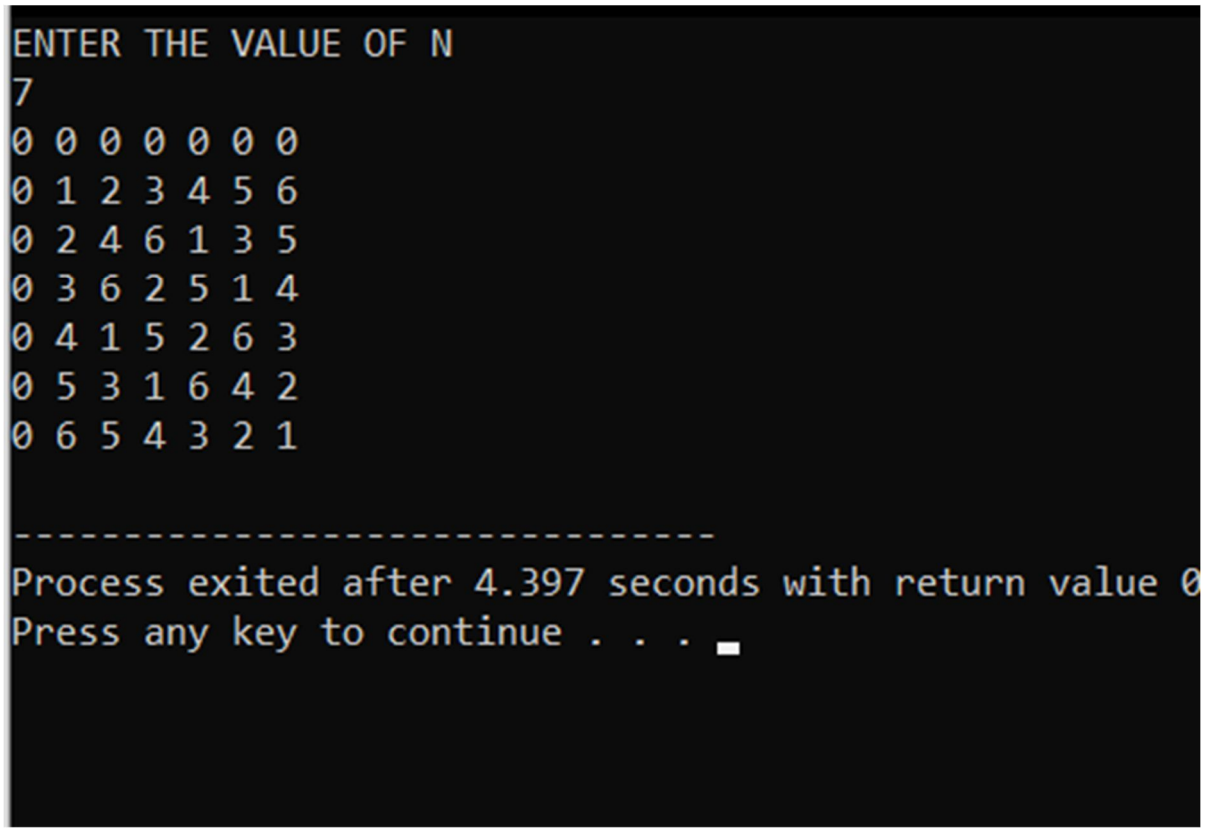
The code basically takes input of a number n from the user. The code generalized to accept any value of n and calculate the result.

Then to calculate binary operation table for n we run two loops from 0 to n-1.

To calculate the value at i th row and j th column we have used the formula (i*j) %n.

```
#include<bits/stdc++.h>

using namespace std.

int main ()

{long long int n, i, j;

cin>>n;

for (i=0; i<=n-1; i++)

        {for (j=0; j<=n-1; j++)

                {cout<<((i*j) %n)<<" ";

                }

                cout<<endl;

        }

        return 0;}
```

CODE RESULT WHEN WE RUN IT FOR THE VALUE OF N=7



Show that (Z,(*)) is a semigroup for any n

SEMIGROUP:

Let us consider, an algebraic system (Z, (*)), where (*) is a binary operation on Z. Then, the system (Z, (*)) is said to be semi-group if it satisfies the following properties:

1. The operation (*) is a closed operation on set Z.
2. The operation (*) is an associative operation.

Example: Consider an algebraic system (Z, *), where Z = {1, 3, 5, 7, 9....}, the set of positive odd integers and * is a binary operation means multiplication. Determine whether (Z, (*)) is a semi-group.

Solution: Closure Property: The operation (*) is a closed operation because multiplication of two +ve odd integers is a +ve odd number.

Associative Property: The operation (*) is an associative operation on set Z. Since every a, b, c ∈ Z, we have

(a * b) * c = a * (b * c)

Hence, the algebraic system (Z, (*)), is a semigroup.

LANGUAGE USED: C++

CODE:

```
#include <iostream>

using namespace std;


bool check_whether_satisfies_closure_property (int n)

{

   cout << "Checking for closure property -->\n";

   cout << "Here (*) denotes the binary operator between the two operands!!\n";


   for (int a = 1; a <= n; a++)

   {

      for (int b = a; b <= n; b++)

      {

         cout << "[" << a << "(*)" << b <<"] =" << (a*b) %n << "\n";

         cout << "[" << b << "(*)" << a <<"] =" << (b*a) %n << "\n";

         if((a*b) %n == (b*a) %n) cout << "lies in the range \n";
```

```
            else
            {
                cout << "Does not lie in the range 0 to n-1 \n";
                cout << "Closure property not satisfied!!\n";
                return false;
            }
        }
    }
    cout << "Closure property satisfied! \n";
    return true;
}
bool check_whether_satisfies_associative_property (int n)
{
    cout << "Checking for associative properly -->\n";
    cout << "Here (*) denotes the binary operator between the two operands!!\n";
    for(int a = 1; a <= n; a++)
    {
        for (int b = 1; b <= n; b++)
        {
            for (int c = 1; c <= n; c++)
            {
                cout << "[[" << a << "(*)" << b << "]" << "(*)" << c << "]  =  " << "["
<< (a*b)%n << "(*)" << c << "]  =  " << (((a*b)%n)*c)%n << "\n";
                cout << "[" << a << "(*)" << "[" <<  b << "(*)" << c << "]  =  " << "["
<< a << "(*)" << (b*c)%n << "]  =  " << (a*((b*c)%n))%n << "\n";
                if(((((a*b) %n) *c) %n) == ((a*((b*c) %n)) %n)) cout << "equal! \n";
                else
                {
```

```
                cout << "Not Equal!!\n";
                cout << "Associative property not satisfied!!\n";
                return false;
            }
        }
    }
    cout << "Associative property satisfied! \n";
    return true;
}

int main ()
{
    int n;
    cout << "Enter the value of n:- ";
    cin >> n;
    bool closure = check_whether_satisfies_closure_property(n);
    cout << "\n";
    bool associative = check_whether_satisfies_associative_property(n);
    cout << "\n\n";
    if (closure & associative)
    {
        cout << "Since it satisfies both closure & associative property\n";
        cout << "Hence it's a Semigroup! \n";
    }
    else
    {
```

```
        if (! closure)
        {
       cout << "Since it doesn't satisfy closure property\n Hence not a Semigroup!!\n";
        }
        else if (! associative)
        {
     cout << "Since it doesn't satisfy associative property\n Hence not a Semigroup!!\n";
        }
        else
        {
        cout << "Since it satisfies neither closure nor associative so not a Semigroup!!\n";
        }
    }
    return 0;
}
```

EXPALNATION OF CODE:

The code basically takes input of a number n from the user. The code generalized to accept any value of n and calculate the result.

For closure property, we have generated all possible pairs in the range from o to n-1 and have checked whether they lie in 0 to n-1 or not.

For associative property, we have generated all triplets in the range – to n-1 and have checked whether they satisfy (((((a(*)b) %n) (*)c) %n) ==((a(*)((b(*)c) %n)) %n)) this or not.

To prove that (Z, (*)) is a semigroup we have to prove that both associative and closure property is satisfied.

OUTPUT:

WHEN THE VALUE OF N =2