# Phase 1

**Code:**

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

class OS
{
private:
    char M[100][4];
    char IR[4];
    char R[4]; // 4 Bit cha general Purpose register
    int IC;
    int SI;
    bool C; // Toggle Register
    char buffer[40];

public:
    void init();
    void LOAD();
    void Execute();
    void MOS();

    fstream infile;
    fstream outfile;
};

// initiallise the memory of OS by setting all value to zero
void OS::init()
{

    for (int i = 0; i < 100; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            M[i][j] = ' ';
        }
    }

    IR[0] = {' '};
    R[0] = {' '};
    C = false;
}

// Master Mode
```

```
void OS::MOS()
{

    if (SI == 1) // Read Mode
    {
        for (int i = 0; i <= 39; i++)
            buffer[i] = '\0';

        infile.getline(buffer, 40);

        int k = 0;
        int i = IR[2] - 48;
        i = i * 10;

        for (int l = 0; l < 10; ++l)
        {
            for (int j = 0; j < 4; ++j)
            {
                M[i][j] = buffer[k];
                k++;
            }
            if (k == 40)
            {
                break;
            }
            i++;
        }

        for (int i = 0; i < 100; i++)
        {
            cout << "M[" << i << "]\t";
            for (int j = 0; j < 4; j++)
            {
                cout << M[i][j];
            }
            cout << endl;
        }
    }
    else if (SI == 2) // Write Mode
    {
        for (int i = 0; i <= 39; i++)
            buffer[i] = '\0';

        int k = 0;
        int i = IR[2] - 48;
        i = i * 10;

        for (int l = 0; l < 10; ++l)
        {
            for (int j = 0; j < 4; ++j)
            {
                buffer[k] = M[i][j];
                outfile << buffer[k];

                k++;
            }
            if (k == 40)
            {
                break;
            }
            i++;
        }
        for (int i = 0; i < 100; i++)
        {
            cout << "M[" << i << "]\t";
            for (int j = 0; j < 4; j++)
            {
                cout << M[i][j];
            }
            cout << endl;
        }

        outfile << "\n";
    }
    else if (SI == 3) // Terminate
    {

        // outfile << "\n";
        cout << "Code is Terminated Successfully!!" << endl;
        outfile << "\n";
    }
}

// Execution
// This function executes the program that has been loaded into main
memory, decodes them & executes them
void OS::Execute()
{
    while (true)
    {
        for (int i = 0; i < 4; i++) // Load in register
        {
            IR[i] = M[IC][i];
        }
        IC++;
```

```
if (IR[0] == 'G' && IR[1] == 'D') // GD
{
    SI = 1;
    MOS();
}
else if (IR[0] == 'P' && IR[1] == 'D') // PD
{
    SI = 2;
    MOS();
}
else if (IR[0] == 'H') // H
{
    SI = 3;
    MOS();
    break;
}
else if (IR[0] == 'L' && IR[1] == 'R') // LR
{
    int i = IR[2] - 48;
    i = i * 10 + (IR[3] - 48);

    for (int j = 0; j <= 3; j++)
        R[j] = M[i][j];

    // for(int j=0;j<=3;j++)
    //  cout<<R[j];

    cout << endl;
}
else if (IR[0] == 'S' && IR[1] == 'R') // SR
{
    int i = IR[2] - 48;
    i = i * 10 + (IR[3] - 48);
    // cout<<i;
    for (int j = 0; j <= 3; j++)
        M[i][j] = R[j];

    cout << endl;
}
else if (IR[0] == 'C' && IR[1] == 'R') // CR
{
    int i = IR[2] - 48;
    i = i * 10 + (IR[3] - 48);
    // cout<<i;
    int count = 0;

    for (int j = 0; j <= 3; j++)
        if (M[i][j] == R[j])
```

```
                    count++;

                if (count == 4)
                    C = true;

                // cout<<C;
            }
            else if (IR[0] == 'B' && IR[1] == 'T') // BT
            {
                if (C == true)
                {
                    int i = IR[2] - 48;
                    i = i * 10 + (IR[3] - 48);

                    IC = i;
                }
            }
        }
}

// Load Function
// Loads a program into the main memory of OS
void OS::LOAD()
{

    cout << "Reading Data..." << endl;
    int x = 0;
    do
    {
        for (int i = 0; i <= 39; i++) // clear buffer
            buffer[i] = '\0';

        infile.getline(buffer, 40);

        for (int k = 0; k <= 39; k++)
            cout << buffer[k];

        if (buffer[0] == '$' && buffer[1] == 'A' && buffer[2] == 'M' &&
buffer[3] == 'J')
        {
            init();
        }
        else if (buffer[0] == '$' && buffer[1] == 'D' && buffer[2] ==
'T' && buffer[3] == 'A')
        {
            IC = 00;
            Execute();
        }
        else if (buffer[0] == '$' && buffer[1] == 'E' && buffer[2] ==
'N' && buffer[3] == 'D')
        {
            x = 0;
            continue;
        }
        else
        {
            int k = 0;

            for (; x < 100; ++x)
            {
                for (int j = 0; j < 4; ++j)
                {

                    M[x][j] = buffer[k];
                    k++;
                }

                if (k == 40 || buffer[k] == ' ' || buffer[k] == '\n')
                {
                    break;
                }
            }
        }

    } while (!infile.eof()); // continues to take input till eof
}

int main()
{
    OS os;

    os.infile.open("input.txt", ios::binary | ios::in);
    os.outfile.open("output.txt", ios::binary | ios::out);

    if (!os.infile)
    {
        cout << "Failure" << endl;
    }
    else
    {
        cout << "File Exist" << endl;
    }

    os.LOAD();

    return 0;
```

```
}
```

## Input:

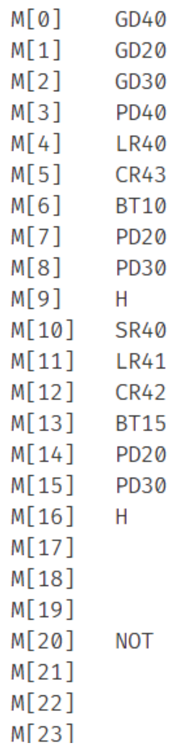
```
$AMJ000100030001
GD10PD10H
$DTA
Hello World!!
$END0001
$AMJ0002000120004
GD20GD30GD40GD50PD20PD30LR20CR30BT10
PD40PD50H
$DTA
VIT
VIT
NOT SAME
IS SAME
$END0002
$AMJ0005000170003
GD40GD20GD30PD40LR40CR43BT10PD20PD30
H   SR40LR41CR42BT15PD20PD30H
$DTA
aaaaabbbbbbaaaab
NOT
PALINDROME
$END0005
```

## Output in Text File:

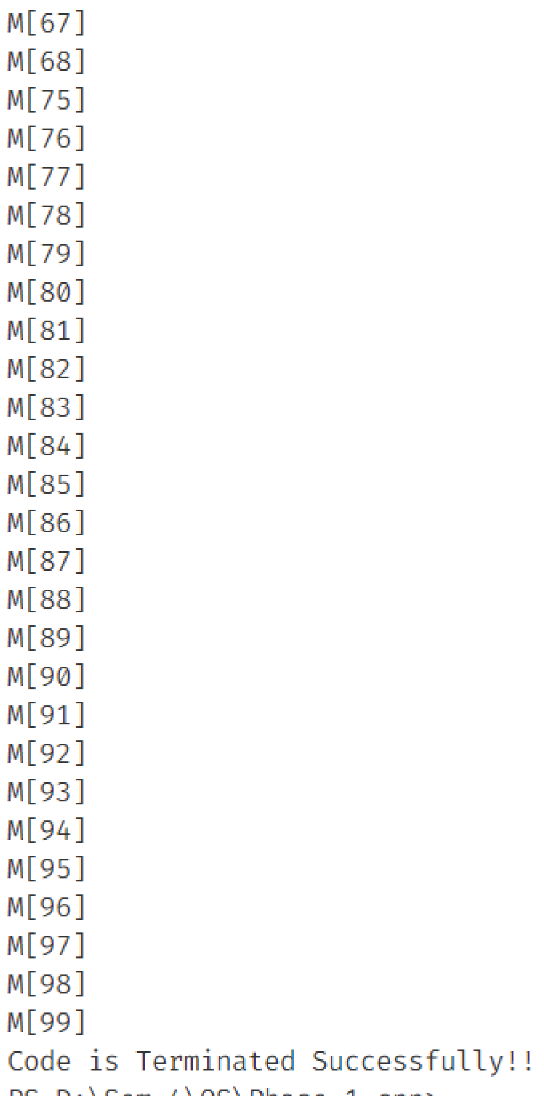
```
Hello World!!


VIT

VIT

IS SAME


aaaaabbbbbbaaaab

NOT

PALINDROME
```

**Output on Terminal:**

```
M[0]SR40GD40CR42BT15PD20PD30HD20PD30
M[1]    GD20
M[2]    GD30
M[3]    PD40
M[4]    LR40
M[5]    CR43
M[6]    BT10
M[7]    PD20
M[8]    PD30
M[9]    H
M[10]   SR40
M[11]   LR41
M[12]   CR42
M[13]   BT15
M[14]   PD20
M[15]   PD30
M[16]   H
M[17]
M[18]
M[19]
M[20]
M[21]
M[22]
M[23]
M[24]
M[25]
M[26]
```



```
M[0]    GD40
M[1]    GD20
M[2]    GD30
M[3]    PD40
M[4]    LR40
M[5]    CR43
M[6]    BT10
M[7]    PD20
M[8]    PD30
M[9]    H
M[10]   SR40
M[11]   LR41
M[12]   CR42
M[13]   BT15
M[14]   PD20
M[15]   PD30
M[16]   H
M[17]
M[18]
M[19]
M[20]   NOT
M[21]
M[22]
M[23]
```



```
M[67]
M[68]
M[75]
M[76]
M[77]
M[78]
M[79]
M[80]
M[81]
M[82]
M[83]
M[84]
M[85]
M[86]
M[87]
M[88]
M[89]
M[90]
M[91]
M[92]
M[93]
M[94]
M[95]
M[96]
M[97]
M[98]
M[99]
Code is Terminated Successfully!!
```