```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm> // Para std::reverse

using namespace std;

// Enumeración para la dirección del movimiento del cabezal
enum Direccion { IZQ, DER };

// Estructura para definir cada transición de la máquina
struct Transiciones {
    string estadoActual;
    char simboloLectura;
    string estadoSiguiente;
    char simboloEscritura;
    Direccion dirMov;
};

// Clase que representa la Máquina de Turing
class MaquinaTuring {
private:
    string estadoActual;
    int cabezal;
    vector<char> cinta;
    vector<Transiciones> transiciones;

public:
    // Constructor de la máquina
    MaquinaTuring(string estadoInicial, vector<Transiciones> tr, string entrada)
        : estadoActual(estadoInicial), transiciones(tr), cabezal(0) {

        // La cinta inicia con un símbolo en blanco para la computación
        cinta.push_back('_');
        for (char const &c : entrada) {
            cinta.push_back(c);
        }
        // Se asegura de que haya espacio para trabajar al final
        cinta.push_back('_');
        cinta.push_back('_');
    }

    // Método para ejecutar la máquina hasta que se detenga
    void ejecutar() {
        while (true) {
            bool transicionEncontrada = false;

            // Imprime el estado actual para depuración (opcional)
            // cout << "Estado: " << estadoActual << ", Cabezal en: " <<
cinta[cabezal] << endl;

            for (const auto& t : transiciones) {
                if (t.estadoActual == estadoActual && t.simboloLectura ==
cinta[cabezal]) {
                    // Aplica la transición
                    cinta[cabezal] = t.simboloEscritura;
                    estadoActual = t.estadoSiguiente;

                    // Mueve el cabezal
```

```
                    cabezal += (t.dirMov == DER) ? 1 : -1;

                    // Asegura que el cabezal no se salga de la cinta por la
izquierda
                    if (cabezal < 0) {
                        cinta.insert(cinta.begin(), '_');
                        cabezal = 0;
                    }
                    // Asegura que haya espacio si el cabezal llega al final
                    if (cabezal >= cinta.size()) {
                        cinta.push_back('_');
                    }

                    transicionEncontrada = true;
                    break;
                }
            }
            // Si no se encuentra ninguna transición, la máquina se detiene
            if (!transicionEncontrada) {
                break;
            }
        }

        // Imprime el resultado final
        cout << "Estado final: " << estadoActual << endl;
        cout << "Cinta final: ";
        for (char c : cinta) {
            cout << c;
        }
        cout << endl;
    }
};

int main() {
    // Definición de las transiciones basadas en el diagrama
    vector<Transiciones> trs = {
        // --- GRUPO "MULTIPLICA POR 2" ---
        // q0: Mueve el cabezal a la derecha hasta el final de la entrada
        {"q0", '0', "q0", '0', DER},
        {"q0", '1', "q0", '1', DER},
        {"q0", '_', "q1", '_', IZQ},

        // q1 y q2: Parte del proceso de multiplicación (interpretación del
diagrama)
        {"q1", '1', "q1", '0', IZQ},
        {"q1", '0', "q2", '0', IZQ},
        {"q1", '_', "q4", '_', DER}, // Salta al grupo "SUMA 1"

        {"q2", '1', "q2", '1', IZQ},
        {"q2", '0', "q1", '1', IZQ},
        // Se omite la transición a q3 ya que su lógica es ambigua en el diagrama




        // --- GRUPO "SUMA 1" ---
        // q4: Mueve el cabezal a la derecha hasta el final del número
        {"q4", '0', "q4", '0', DER},
```

```
        {"q4", '1', "q4", '1', DER},
        {"q4", '_', "q5", '_', IZQ},

        // q5: En el último dígito, comienza la suma
        {"q5", '0', "q6", '1', IZQ},
        {"q5", '1', "q6", '1', IZQ}, // Nota: Diagrama ambiguo, se asume que ambos
0 y 1 van a q6
        {"q5", '_', "q7", '1', IZQ}, // Caso de acarreo (ej. entrada "111")

        // q6: Proceso de acarreo (carry)
        {"q6", '0', "q6", '0', IZQ},
        {"q6", '1', "q6", '0', IZQ},
        {"q6", '_', "q5", '_', DER},

        // q7: Manejo del acarreo final y bucles
        {"q7", '1', "q7", '0', IZQ},
        // La máquina se detendrá en este punto si lee un '0' o '_'
        // ya que no hay transiciones definidas para ellos desde q7.
    };

    string entrada;
    cout << "Esta maquina calcula Y = 2X + 1 para un numero binario X." << endl;
    cout << "Ingrese el numero binario (X): ";
    cin >> entrada;

    string estadoInicial = "q0";

    // Crear y ejecutar la máquina de Turing
    MaquinaTuring mt(estadoInicial, trs, entrada);
    mt.ejecutar();

    return 0;
}
```