Name & Title: - Quantum Computing Resource Management System

Introduction :- A Quantum Computing Resource Management System (QCRMS) is a software framework designed to manage the allocation and utilisation of quantum computing resources. With the rise of quantum computing, QCRMS has become increasingly important for research and commercial organisations seeking to leverage quantum computing power. The system optimises the use of available resources, manages workload scheduling, and ensures efficient execution of quantum computing tasks. It also enables resource sharing, collaboration, and access control. In this way, QCRMS provides a centralised solution for managing the complexities of quantum computing resource allocation and usage.

Enums Used:

Name	Description
SINGLE_QUBIT_GATE	Represents a gate that operates on a single qubit. Examples of such gates include the Pauli gates (X, Y, Z) and the Hadamard gate.
TWO_QUBIT_GATE	Represents a gate that operates on two qubits. Examples of such gates include the CPHASE gate and the Controlled-NOT (CNOT) gate.
MEASUREMENT_GATE	Represents a gate that measures the state of a qubit.
CNOT_GATE	Represents a two-qubit gate that performs a conditional NOT operation, where the target qubit is flipped if and only if the control qubit is in the state
SWAP_GATE	Represents a two-qubit gate that swaps the state of two qubits.

Structures Used:

Gate:

Data Member	Description
qubitIndex	An integer value representing the index of the qubit on which the gate is applied
gateType	An enumerated type representing the type of gate applied to the qubit

Quantum Circuit:

Data Member	Description
numQubits	An integer value representing the number of qubits in the quantum circuit.
qubitStates	A pointer to an array of integers, where each element represents the state of a qubit (either 0 or 1) in the quantum circuit.
gates	A pointer to an array of Gate structures, representing the quantum gates applied in the circuit.
numGates	An integer value representing the number of gates applied in the circuit.

Problem Description:

The following functions will be used for the assignment:

Function Name	Parameter	Return Type	Description
createQuantumCircuit	int numQubits	QuantumCircuit*	Allocates memory for QuantumCircuit struct and initialise its fields like numQubits, numGates, qubitStates, and gates.
addGateToCircuit	QuantumCircuit *circuit, GateType gateType, int qubitIndex	void	Adds a single qubit gate, measurement gate or two qubit gate to the circuit based on the gateType and qubitIndex.
destroyQuantumCircuit	QuantumCircuit *circuit	void	Deallocates memory for QuantumCircuit struct and frees any dynamically allocated memory inside the Gate struct.
applySingleQubitGate	int qubitState, GateType gateType, QuantumCircuit *circuit	int	Applies a single qubit gate to the qubitState of the given circuit based on the gateType.
applyTwoQubitGate	qubitState1: int, qubitState2: int, gateType: GateType, circuit: QuantumCircuit*	int	Apply a two-qubit gate (CNOT or SWAP) to the specified qubits in the given quantum circuit. Returns 0 if successful, or a negative integer code indicating the specific error encountered.
measureQubit	circuit: QuantumCircuit*, qubitIndex: int	int	Measure the specified qubit in the given quantum circuit. Returns the measurement result (0 or 1) if successful, or a negative integer code indicating the specific error encountered.

Test Cases :

Test Case Name	Input	Output	Explanation
testCreateQuantumCircuit	3	Create a QuantumCircuit with 3 qubits, assert that the circuit is not null, numQubits is 3, numGates is 0, and qubitStates and gates are not null, and destroy the circuit.	Test whether createQuantumCircuit function creates the QuantumCircuit object with the given number of qubits and initialises its properties correctly.
testCreateQuantumCircuit2	5	Create a QuantumCircuit with 5 qubits, assert that circuit is not null, numQubits is 5, numGates is 0, qubitStates are all 0, gates are initialised as SINGLE_QUBIT_GATE, and destroy the circuit.	Test whether createQuantumCircuit function creates the QuantumCircuit object with the given number of qubits and initialises its properties correctly.
testApplySingleQubitGate	QuantumCircuit with 1 qubit	Assert that the qubitState is updated correctly after applying SINGLE_QUBIT_GATE and TWO_QUBIT_GATE, and destroy the circuit.	Test whether applySingleQubitGate function applies the SINGLE_QUBIT_GATE or TWO_QUBIT_GATE on the given qubit state, and returns the updated state.
testAddGateToCircuit	QuantumCircuit with 2 qubits, CNOT_GATE on qubit 0, and TWO_QUBIT_GATE on qubit 1	Assert that the gate is added correctly to the circuit, the circuit's numGates is incremented, and the added gates have the correct properties, and destroy the circuit.	Test whether addGateToCircuit function adds the given gate to the circuit and updates the circuit's properties correctly.
testAddSingleQubitGate	QuantumCircuit with 1 qubit, SINGLE_QUBIT_GATE on qubit 0	Assert that the gate is added correctly to the circuit, the circuit's numGates is incremented, and the added gate has the correct properties, and destroy the circuit.	Test whether addGateToCircuit function adds the SINGLE_QUBIT_GATE to the circuit and updates the circuit's properties correctly.
testAddTwoQubitGate	QuantumCircuit with 2 qubits,	Assert that the gate is added correctly to the circuit, the circuit's	Test whether addGateToCircuit function adds the

	TWO_QUBIT_GATE on qubit 0	numGates is incremented, and the added gates have the correct properties, and destroy the circuit.	TWO_QUBIT_GATE to the circuit and updates the circuit's properties correctly.
testDestroySingleQubitGate	QuantumCircuit with 1 qubit, SINGLE_QUBIT_GATE on qubit 0	Assert that the circuit is destroyed and the circuit is null.	Test whether destroyQuantumCircuit function destroys the given circuit and sets it to null.
testDestroyCircuitWithTwoQubitGates	QuantumCircuit with 2 qubits, TWO_QUBIT_GATE on both qubits	Assert that the circuit is destroyed and the circuit is not null.	Test whether destroyQuantumCircuit function destroys the given circuit with two qubit gates and sets it to null.
testDestroyCircuitWithMeasurementGates	None	None	This test case checks if the QuantumCircuit is destroyed properly when it contains measurement gates.
testApplySingleQubitGate_ValidInputs_Success	None	None	This test case applies a single qubit gate to a QuantumCircuit with valid inputs, and checks if the qubit states, number of gates and the applied gate are updated properly.
testApplySingleQubitGate_InvalidCircuit_ReturnsError	None	-1	This test case checks if the applySingleQubitGate function returns -1 when a NULL QuantumCircuit is provided.
testApplySingleQubitGate_InvalidQubitIndex_ReturnsError	None	-2	This test case checks if the applySingleQubitGate function returns -2 when an invalid qubit index is provided.
testApplyTwoQubitGate	None	-3	This test case checks if the applyTwoQubitGate function returns -3 when the provided gate is a CNOT gate and the target qubit is not in the

testInvalidCircuit	None	-1	This test case checks if the applyTwoQubitGate function returns -1 when a NULL QuantumCircuit is provided.
testInvalidQubitIndex	None	-2	This test case checks if the applyTwoQubitGate function returns -2 when an invalid qubit index is provided.
testMeasureQubit	None	None	This test case measures a qubit in a QuantumCircuit with one qubit and checks if the measurement result is either 0 or 1.
testMeasureQubitInvalidIndex	None	-1,-1	This test case checks if the measureQubit function returns -1 when an invalid qubit index is provided, both negative and exceeding the maximum qubit index.
testMeasureQubitSuccess	None	None	This test case measures a qubit in a QuantumCircuit with one qubit and checks if the measurement result is equal to the qubit state, and the number of gates is updated properly