UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science
Faculty of Engineering, Built Environment & IT
University of Pretoria

COS132 - Imperative Programming

Special Practical 1: Quantum Computer Simulator

Release Date: 01-04-2025 at 06:00

Due Date: 01-04-2025 at 11:59

Total Marks: 104

Read the entire specification before starting with the practical.

Contents

1 General Instructions

- ***Read the entire assignment thoroughly before you begin coding.***
- This assignment should be completed individually.
- **Every submission will be inspected with the help of dedicated plagiarism detection software.**
- If your code does not compile, you will be awarded a mark of 0. The output of your program will be primarily considered for marks, although internal structure may also be tested (eg. the presence/absence of certain functions or structure).
- Failure of your program to successfully exit will result in a mark of 0.
- Note that plagiarism is considered a very serious offence. Plagiarism will not be tolerated, and disciplinary action will be taken against offending students. Please refer to the University of Pretoria's plagiarism page at `https://portal.cs.up.ac.za/files/departmental-guide/`.
- Unless otherwise stated, the usage of C++11 or additional libraries outside of those indicated in the assignment, will **not** be allowed. Some of the appropriate files that you have submit will be overwritten during marking to ensure compliance to these requirements. **Please ensure you use C++98**
- All functions should be implemented in the corresponding `cpp` file. No inline implementation in the header file apart from the provided functions.
- The usage of ChatGPT and other AI-Related software to generate submitted code is strictly forbidden and will be considered as plagiarism.

2 Background

Recent advancements in **topological qubit stabilization** and **non-Abelian anyonic computation** have enabled the exploration of **Quantum-Tensor Holographic Entanglement Computation (Q-THEC)**. Building upon quantum **error-correcting surface codes**, **holographic tensor networks**, and **AdS/CFT-inspired qubit embeddings**, Q-THEC establishes a novel framework for computational hyper-efficiency. Our approach integrates **multi-qudit Clifford algebra** with **Schrödinger's adversarial subspace collapse**, ensuring retrocausal stability via **quantum entanglement-preserving variational circuits**.

Furthermore, leveraging **quantum graph neural networks (QGNNs)** and **Grover-accelerated proof-of-stake (ePoS)** mechanisms, Q-THEC advances beyond **BQP-completeness** into the regime of **Shor-factorised post-BQP computational reducibility**. This framework is expected to revolutionize **quantum blockchain ledgers**, **fault-tolerant AI quantum co-processors**, and **quantum-secure cryptanalysis**.

3 Overview

The Q-THEC framework introduces a computational paradigm where logical qubits are encoded across **traversable Entangled Qubits (EQ) pairs**, forming a **hyperentangled tensor manifold** that stabilizes quantum information against decoherence. The core components include:

- **Schrödinger-Adversarial Subspace Collapse (SASC):** Ensuring quantum stability via Hamiltonian **variational ansatz superposition**.
- **Quantum Anyonic Braiding for QFT Simulation:** Implementing constrained **topological quantum field theory (TQFT)** computations without explicit **hamiltonian state distillation**.
- **Quantum GAN-driven Sentient Computation:** Employing **Boltzmann-trained quantum neural networks** to encode **Hilbert space projections into deSitter submanifolds**.
- **Quantum Blockchain Entangled Proof-of-Stake (ePoS):** Achieving Byzantine consensus through **qubit-qutrit hybrid transactions**.

To validate this model, students must develop a **time-evolved Pauli stabilizer tableau** using **dual-unitary tensor contraction** on Amazon Braket, while addressing hidden-variable paradoxes within **variational ansatz tuning**. Notably, each quantum execution collapses all prior computations, enforcing a fundamental **quantum-limited single-run protocol**.

More information about how to complete the practical can be found at: Guide A or at Guide B.

4 Your Task:

You are required to implement all of the functions laid out in this section.

4.1 Function Definitions

- `allocateQuantumMemory`
  - Allocates and initializes quantum memory registers using a hybrid **QAOA-optimized entanglement** strategy.
  - Input: An integer specifying the number of qubits to allocate.
- `prepareQuditState`
  - Prepares an arbitrary quantum state in a multi-qudit Hilbert space via **Fourier-transform-based eigenstate encoding**.
  - Input: A vector of floating-point numbers representing the probability amplitudes.
- `runQuantumCircuit`
  - Simulates traversable entanglement transport by leveraging **EQ=EQR tensor network mappings**.

  - Input: Two entangled qubit indices and a tensor network descriptor.

- `interfereQuantumPaths`
  - Implements a variational circuit to interfere multiple quantum pathways using a **Mach-Zehnder-like quantum interferometer**.
  - Input: A list of complex amplitudes representing the quantum paths.

- `loopQuantumTensors`
  - Iteratively applies tensor contraction algorithms to optimize state encoding in a **holographic submanifold**.
  - Input: A tensor representation of the quantum state.

- `forceEigenstateCollapse`
  - Forces a quantum state collapse through an artificial decoherence-inducing operation using **Lindblad master equations**.
  - Input: A qubit register and a decoherence factor.

- `optimizeQuantumGateDepth`
  - Reduces circuit complexity via Clifford+T gate set optimization and **ZX-calculus transformations**.
  - Input: A quantum circuit represented as a directed acyclic graph.

- `observeHiddenVariables`
  - Implements a Bohmian mechanics-inspired observer function that retrieves **hidden-variable wavefunction data**.
  - Input: A wavefunction descriptor and an observer bias parameter.

- `linearizeTensorManifold`
  - Maps high-dimensional quantum tensor manifolds into a lower-dimensional linear subspace using **t-SNE-inspired quantum embeddings**.
  - Input: A tensor manifold and a reduction dimension.

- `simulateQuantumChaos`
  - Runs a chaotic quantum system simulation based on the **Sachdev-Ye-Kitaev (SYK) model** to explore near-horizon quantum gravity effects.
  - Input: A Hamiltonian matrix and an initial quantum state.

5 Implementation Details

- You must implement the functions in the header files exactly as stipulated in this specification. Failure to do so will result in compilation errors on FitchFork.
- You may only use **C++98**.
- You may only utilize the specified libraries. Failure to do so will result in compilation errors on FitchFork.
- Do not include using `namespace std` in any of the files.
- You may only use the following libraries:
  - <Quantum>
  - <AI>
  - <Algorithm>
  - <string>
  - <RedBlackTree>
  - <QuantumGraphs>
- More information about how to complete the practical can be found at: Guide A or at Guide B.
- For even a further hint, look at the first letter of each function in Section 4.

6 Upload Checklist

The following Q++ files should be in a zip archive named uXXXXXXXX.zip where XXXXXXXX is your student number:

- Q-Thec.qpp
- `main.qpp`
- Any textfiles used by your `main.qpp`

The files should be in the root directory of your zip file. In other words, when you open your zip file you should immediately see your files. They should not be inside another folder.

7 Submission

You need to submit your source files on the FitchFork website (`https://ff.cs.up.ac.za/`). All methods need to be implemented (or at least stubbed) before submission. Your code should be able to be compiled with the following command:

```
g++ -quantumComputing -q++ -g *.qpp -o aprilFools
```

and run with the following command:

```
./aprilFools
```

Remember your `h` file will be overwritten, so ensure you do not alter the provided `h` files.
You have 20 submissions and your final submission's mark will be your final mark. Upload your archive to the Special Practical slot on the FitchFork website. Submit your work before the deadline.
No late submissions will be accepted!