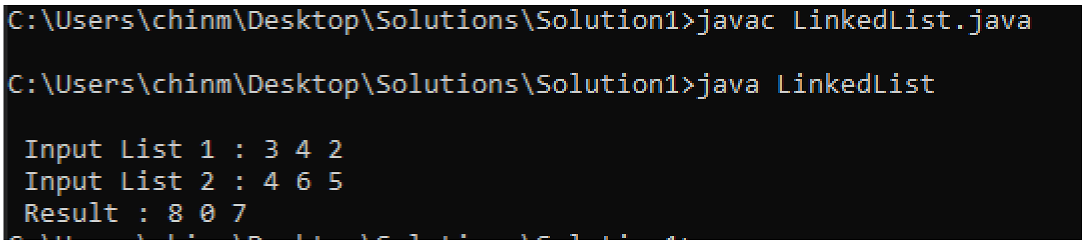
1. Add Two Numbers
You are given two non-empty linked lists representing two non-negative integers.
The digits are stored in reverse order, and each of their nodes contains a single digit.
Add the two numbers and return the sum as a linked list.
You may assume the two numbers do not contain any leading zero, except the number 0 itself.
Answer:
Please find the attached ‘LinkedList.java’ file for code.
output:

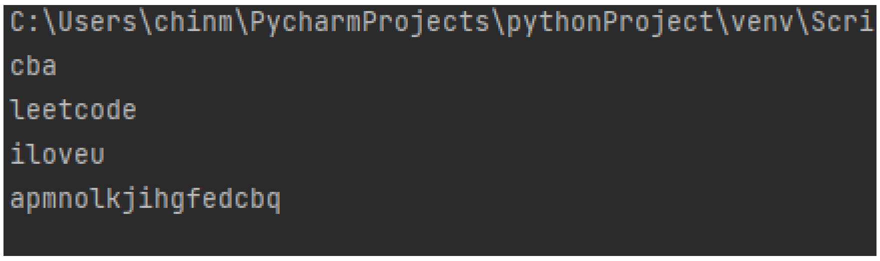


```
C:\Users\chinm\Desktop\Solutions\Solution1>javac LinkedList.java

C:\Users\chinm\Desktop\Solutions\Solution1>java LinkedList

 Input List 1 : 3 4 2
 Input List 2 : 4 6 5
 Result : 8 0 7
```

2. `Reverse string`
Answer:
Please find the attached ‘ReverseString.py’ file for code.
output:



```
C:\Users\chinm\PycharmProjects\pythonProject\venv\Scri
cba
leetcode
iloveu
apmnolkjihgfedcbq
```

3. Prometheus & Grafana
https://prometheus.io/docs/visualization/grafana/
We want to use Prometheus & Grafana to monitor devices in the field.
If we asked you to set this up what do you need?
Answer:
Information needed: Number of machines we need to monitor, IP address of all Machines, which machine can be used as a server, need to create scripts to install and run server and clients for Prometheus & Grafana

Steps:
1. Make one device as a server for Prometheus & Grafana,
Create scripts for to install Prometheus & Grafana server client
2. Install Prometheus node exporter agent on each device, it ll create and start service
Also It ll provide endpoint which will expose the metrics
3. Install Prometheus Server,
Create yml file which ll contain list of job names and endpoints for all client devices
4 Start Prometheus Server,
Hit the server end point for Prometheus Server
For e.g (10.0.0.3:9090/targets)
5. Install and Launch Grafana on server
6. Add data source Prometheus

7. create or import dashboard

5. As a test engineer, what do you expect you to receive as input?
Answer:
1. Test Strategy
2. Test management, Requirement management, Automation tool and bug reporting tools
3. Clearly defined set of requirements/ Acceptance criteria
4. Frontend Design blueprint
5. Test environment setup (For e.g web application, mobile application under test should be deployed

6. As a test engineer, what would you generate as a result of output?
Answer:
1. Draft Test cases which cover all acceptance criterias and test scenarios (Happy path test cases)
2. Write positive, negative, boundary and end to end test cases
3. Create Test plans for each build. Smoke test plan and Regression test plan
4. Manually test above test plans for each build
5. Track bugs in bug reporting tool (jira) and track root cause escalate them to Dev
6. Add new test cases for newly developed functionalities
7. Automate test cases to save time and manual efforts
8. Execute Automation test suite and maintain it

Concept Questions
=================
7. Describe the software development life cycle:
Answer:
A software development life-cycle is the process where a software is conceptualized, built, deployed and maintained. The typical phases in software development life-cycle are:
1. Planning - Once a software is conceptualized, its development process is planned to define its development timeline and resources required.
2. Requirement gathering - This phase involves gathering the functional requirements of the proposed software and defining what is necessary to develop them.
3. Designing - In this phase, the user interface and experience is designed.
4. Building - This phase involves writing of the actual code and is the most important of all.
5. Documenting - In this phase, all the relevant details of the software are documented for future development and for users.
6. Testing - In this phase, the code is verified against its functional requirements.
7. Deployment - Once a software is thoroughly tested, it can be deployed to users to use.

8. Maintaining - This phase involves updating the code to reduce bugs or add new functionalities.

7a. What are the documentations that follow each phase?
Answer:
The following documentation follows each phase:
1. Planning - This phase is followed by creation of high level documents related to development timeline and major functions.
2. Requirement gathering - This phase is followed by generating detailed documents of major and minor requirements of the software.
3. Designing - This phase is followed by documenting the software UI and UX workflows.
4. Building - This phase documents class diagrams, state diagrams, sequence diagrams, code comments, etc..
5. Documenting - This phase includes documenting user guides, installation guides, and other system documentation.
6. Testing - This phase documents test plans, bugs reported, bugs fixed, and overall software quality.
7. Deployment - This phase is followed by documenting what needs to be improved in the software.
8. Maintaining - Typical documentation in this phase includes new functionality documentation and bug fixes.

8. What do you think your day-to-day tasks would be?
Answer:
1. Work with product and software dev. teams to understand software functionalities, and user stories.
2. Perform software/hardware integration tests, regression tests, along with other tests to ensure the intended functionalities are captured in the software.
3. Automate workflows, regression tests, and facilitate CI/CD.
4. Root-cause and track bugs and ensure the correct bug fixes are integrated into the software.

9. If you join the team, at the end of 1 Month, 1 Quarter, 1 year what do you think are quantitative goals that measure your impact?
Answer:
1. Goals in 1 month:
- Learn the product
- Learn requisite technologies.
- Become familiar with the team and culture.
- Independently Deliver on assigned tasks

2. Goals in 1 Quarter:

- Identify 1-2 key areas of improvement.
- Work on solutions to fix them.

3. Goals in 1 year:

- Lean 2-3 new skills required to have a higher impact.
- Identify areas of large scale improvement in workflow/process efficiency.
- Implement solutions to improve the same.