1. Create an API that lists the title, description based on the category passed as an input parameter.

   Ans:-

```
#include <cppcms/application.h>
#include <cppdb/frontend.h>

class ListingsApi : public cppcms::application {
public:
  ListingsApi(cppcms::service &srv) : cppcms::application(srv)
  {
    dispatcher().assign("/listings/(\\w+)", &ListingsApi::get_listings, this, 1);
  }

  void get_listings(std::string category)
  {
    cppdb::session sql("sqlite:db=listings.db");
    cppdb::result res = sql << "SELECT title, description FROM listings WHERE category = ?"
<< category;
    response().out() << "[";
    bool first = true;
    while(res.next()) {
      if(!first) {
        response().out() << ",";
      }
      first = false;
      response().out() << "{\"title\":\"" << res.get<std::string>("title") <<
"\",\"description\":\"" << res.get<std::string>("description") << "\"}";
    }
    response().out() << "]";
  }
};
```

2. Create an API that would save a new entry with all the relevant properties which retrieves values from the endpoint GET /entries.

Ans:- from flask import Flask, jsonify, request

```
app = Flask(__name__)

entries = []

@app.route('/entries', methods=['GET', 'POST'])
def handle_entries():
    if request.method == 'POST':
        data = request.get_json()
        new_entry = {
            'title': data['title'],
            'content': data['content'],
            'author': data['author'],
            'date_published': data['date_published']
        }
        entries.append(new_entry)
        return jsonify(new_entry), 201
    elif request.method == 'GET':
        return jsonify(entries)

if __name__ == '__main__':
    app.run(debug=True)
```

Question3: what are the key things you would consider when creating/consuming an API to ensure that it is secure and reliable?

Using secure communication protocols such as HTTPS

Implementing proper authentication and authorization mechanisms

Input validation and sanitization to prevent against common web vulnerabilities such as SQL injection and cross-site scripting (XSS)

Regularly testing and monitoring the API for security vulnerabilities

When consuming an API, some key things to consider to ensure reliability include:

Checking the API's uptime and availability before integrating it into your application

Implementing error handling and retry logic to handle temporary failures

Caching responses to reduce the load on the API and improve the response time for the client

Monitoring the API's usage to ensure that you are not exceeding any usage limits or quotas.

Suppose you have a CSV file with the data below.

A1: 5, A2: 7, A3: 9, B1: 3, B2: 8, B3: =4+5, C1: =5+A1, C2: =A2+B2, C3: =C2+B3

This can be represented in an excel sheet below:

A B C

1 5 3 =5+A1

2 7 8 =A2+B2

3 9 =4+5 =C2+B3

I want a program that will take the CSV input above and produce CSV output with the results. If it is a value, then return a value. If it is a formula then calculate the formula and return the value of that formula.

1. How will you tackle the challenge above?

2. What type of errors you would you check for?

3. How might a user break your code?

Formulas that contain syntax errors.

Attempts to divide by zero or other mathematical errors when evaluating the formulas.

A user may break the code by:

Providing an incorrectly formatted input CSV file.

Using formulas that reference cells that do not exist or are not in the correct format.

Using formulas that contain syntax errors or other invalid characters.

Attempting to divide by zero or other mathematical errors in the formulas.

Attempting to evaluate malicious code.