**Question number 1**

What are microtasks? What is a microtask queue? What is their role in Promises and how are they different from callbacks?

**Microtasks:** These are the tasks that encounter while the javascript engine is executed the another task. These tasks follow the concept of Asynchronous javascript. The Execution of microtasks can not impact the another statement which are below the microtasks.

**Microtask Queue**: The set of microtask that are enqueued during the javascript engine execution it is called microtask queue. The queue is first-in-first-out: tasks enqueued first are run first.

**Role in Promises:**

**promise . then ( handler );**

**...**

**alert ( "code finished" );**

Promise handlers always go through this internal queue.

If there’s a chain with multiple .then/.catch/finally, then every one of them is executed asynchronously. That is, it first gets queued, then executed when the current code is complete and previously queued handlers are finished.

**Microtask vs Call backs:**

Microtask Queue is like the Callback Queue, but Microtask Queue has higher priority. All the callback functions coming through Promises and Mutation Observer will go inside the Microtask Queue.

### Question number 2

Explain with examples how private, protected variables can be implemented in classes and how can they be used in subclasses?

The private modifier specifies that the member can only be accessed in its own class. The protected modifier specifies that the member can only be accessed within its own package (as with *package-private*) and, in addition, by a subclass of its class in another package.

// protected access modifier

#include <bits/stdc++.h>

using namespace std;

class Parent{

protected:

int id\_protected;

};

class Child : public Parent {

public:

void setId(int id)

{

id\_protected = id;

}

void displayId()

{

cout << "id\_protected is: "

<< id\_protected << endl;

}

};

// main function

int main()

{

Child obj1;

obj1.setId(81);

obj1.displayId();

return 0;

}

// private modifier

#include <iostream>

using namespace std;

class Circle {

private:

double radius;

public:

void compute\_area(double r)

{

radius = r;

double area = 3.14 \* radius \* radius;

cout << "Radius is: " << radius << endl;

cout << "Area is: " << area;

}

};

int main()

{

Circle obj;

obj.compute\_area(1.5);

return 0;

}