Part 1

Note: Total grade for this part is 40. Each correct answer is worth 2 points.

1. *How many times does the for loop iterate?:*
   int counter = 0; for(int i = 0; i >=0 && i < 100; i++){ cout << i << endl; }

2. *What is the output of the following code?*

```
#include <iostream>
using namespace std;

void add_to_sum(){
   static int sum = 0;
   sum += 5;
   cout << sum << " ";
}

void reset_sum(){
   int sum = 0;
   sum += 10;
   cout << sum << " ";
}

int main(){
   for(int i = 0; i < 3; i++){
      add_to_sum();
      reset_sum();
   }
   cout << endl;
   return 0;
}
```

3. *What is the output of the following code?*

```
#include <iostream>
using namespace std;

void display(string text) {
   cout << "String: " << text << endl;
}

void display(int num) {
   cout << "Number: " << num << endl;
}
```

```
int main() {
    display("Hello World");
    display(42);
    return 0;
}
```

4. *What is the output of the following code?*

```
#include <iostream>
using namespace std;

class Circle {
public:
    double radius;

    double calculateArea() {
        return 3.14 * radius * radius;
    }
};

int main() {
    Circle circle1;
    circle1.radius = 10.0;
    cout << "Area of Circle1: " << circle1.calculateArea() << endl;

    Circle circle2;
    circle2.radius = 5.0;
    cout << "Area of Circle2: " << circle2.calculateArea() << endl;

    return 0;
}
```

5. *What is the output of the following code?*

```
#include <iostream>
using namespace std;

void reverseArray(int arr[], int size) {
    for (int i = 0; i < size / 2; i++) {
        int temp = arr[i];
        arr[i] = arr[size - 1 - i];
        arr[size - 1 - i] = temp;
    }
}
```

```
int main() {
    int numbers[] = {5, 10, 15, 20, 25};
    reverseArray(numbers, 5);
    for (int n : numbers) {
        cout << n << " ";
    }
    cout << endl;
    return 0;
}
```

6. Fill in the missing parts of the code to define a class named 'Animal' with private attributes: 'species' (string) and 'age' (int). Provide a public constructor and a method to display animal details.

```
#include <iostream>
using namespace std;

class Animal {
    // Define private attributes here

public:
    // Define constructor here

    // Define method to display animal details
    void displayDetails() {
        cout << "Species: " << species << ", Age: " << age << " years" << endl;
    }
};

int main() {
    Animal animal1("Lion", 5);
    animal1.displayDetails();
    return 0;
}
```

7.Fill in the missing parts of the code to implement dynamic polymorphism using a base class Shape and derived classes Circle and Rectangle. Each derived class should override the virtual draw method to print specific information about the shape.

```
#include <iostream>
#include <string>
using namespace std;

class Shape {
```

```
public:
    // Declare a virtual function 'draw'

};

class Circle : public Shape {
    double radius;
public:
    Circle(double rad) : radius(rad) {}
    // Override 'draw' method for Circle
    // Print "Drawing Circle with radius: " followed by the radius

};

class Rectangle : public Shape {
    double length, width;
public:
    Rectangle(double len, double wid) : length(len), width(wid) {}
    // Override 'draw' method for Rectangle
    // Print "Drawing Rectangle with length and width: " followed by the length and width

};

void renderShape(Shape *shape) {
    shape->draw();
}

int main() {
    Circle circle(5.0);
    Rectangle rectangle(4.0, 3.0);

    renderShape(&circle);
    renderShape(&rectangle);

    return 0;
}
```

8. Predict the output of the above code ?

```
#include <iostream>
using namespace std;

int main() {
    int n = 5;
    int sum = 0;
    for (int i = 1; i <= n; i++) {
        sum += i;
    }
    cout << "Sum of numbers from 1 to " << n << " is " << sum << endl;
    return 0;
```

```
}
```

9. What will be the output of the program ?

```
#include <iostream>
using namespace std;

int main() {
   int numbers[] = {2, 4, 6, 8, 10};
   int sum = 0;
   for (int i = 0; i < 5; i++) {
      sum += numbers[i];
   }
   cout << "Sum: " << sum << endl;
   return 0;
}
```

10. What is the output of this program?

```
#include <iostream>
using namespace std;

int main() {
   int number = 5;
   int *pointer = &number;
   *pointer = 10;
   cout << "number = " << number << endl;
   return 0;
}
```

11. C++ achieves polymorphism by using

12. Virtual functions are very powerful since they are bound at

13. Classes that declare virtual functions should always declare a virtual

Part 2

Note: Total grade for this part is 60. Each correct answer is worth 20 points.

14. Refactoring code from using raw pointers ?

```
#include <iostream>
#include <memory>
using namespace std;

class Device {
public:
   Device() { cout << "Device initialized\n"; }
   ~Device() { cout << "Device shut down\n"; }
};

void operateDevice(Device *dev) {
   cout << "Operating device\n";
}

int main() {
   // Refactor this part to use smart pointers
   Device *device = new Device();
   operateDevice(device);
   delete device;
   return 0;
}
```

15. Refactor below code:

Scenario: You have a logging system that currently logs messages to a file. The system now needs to support logging messages to a database and to a network server in addition to the file.

Task:

Refactor the 'Logging' system to support logging messages to a database and a network server, in addition to a file, For simplicity you don't have to implement file logging or logging to database, just print out something on the console.

Apply the Dependency Inversion Principle and the Interface Segregation Principle in your design.

Hint:

- Create an interface for logging (like ILogging), with a method like logMessage.
- Implement this interface in separate classes for each logging method (File, Database, Network).
- Modify the Logger class to depend on the ILogging interface, not on concrete implementations.
- Ensure that the Logger system can easily switch between different logging methods without changing its own code.

Given Code:

```
#include <iostream>
#include <string>
#include <fstream>
using namespace std;

class FileLogger {
public:
   void logToFile(const string& message) {
      ofstream file("log.txt");
      file << "Logging to File: " << message << endl;
      file.close(); // you don't have to implement like this instead just print out something in the console
   }
};

class Logger {
   FileLogger fileLogger;
public:
   void log(const string& message) {
      fileLogger.logToFile(message);
   }
};

int main() {
   Logger logger;
   logger.log("System started.");
   return 0;
}
```