C++ Final Exam Paper - Predictions and Solutions

Question 1: Class Design and Hierarchy (Shape Hierarchy)

a) Implement the Shape Hierarchy with virtual functions. Create classes for Shape, Circle, and Rectangle. Use polymorphism with virtual functions and implement methods to calculate areas for each specific shape.

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

class Shape {
public:
    virtual double area() const = 0;  // Pure virtual function
    virtual void show() { cout << "Shape\n"; }
};

class Circle : public Shape {
    double radius;
public:
    Circle(double r) : radius(r) {}
    double area() const override { return M_PI * radius * radius; }
    void show() override { cout << "Circle\n"; }
};

class Rectangle : public Shape {
    double width, height;
public:
    Rectangle(double w, double h) : width(w), height(h) {}
    double area() const override { return width * height; }
    void show() override { cout << "Rectangle\n"; }
};

int main() {
    Shape* shape1 = new Circle(5);
    Shape* shape2 = new Rectangle(4, 6);
    shape1->show(); cout << "Area: " << shape1->area() << endl;
    shape2->show(); cout << "Area: " << shape2->area() << endl;
    delete shape1;
    delete shape2;
}
```

Question 2: Theory-based Concepts and Explanation

Write precise descriptions of the following concepts, supported by code examples. Use correct terminology and precise language.

```
1. Polymorphism and Virtual Functions
2. Inheritance Types (Single, Multilevel, Multiple)
3. Abstract Classes and Pure Virtual Functions
4. Exception Handling
5. Upcasting and Downcasting
```

Question 3: Template Class and Exception Handling

a) Write a template class to store an array of values of the same type. Include constructors, overloaded operators, and member functions. Handle exceptions for out-of-bound access.

```
template<typename T>
class MyArray {
    T* arr;
    int size;
public:
    MyArray(int s) : size(s) { arr = new T[size]; }
    ~MyArray() { delete[] arr; }
    void set(int index, T value) { if(index >= size || index < 0) throw out_of_range("Index out of range"); arr[index] = value; }
    T get(int index) { if(index >= size || index < 0) throw out_of_range("Index out of range"); return arr[index]; }
    int getSize() { return size; }
};

int main() {
    MyArray<int> arr(5);
    try {
        arr.set(5, 10);  // Exception will be thrown
    } catch (const out_of_range& e) {
        cout << e.what() << endl;
    }
}
```

Question 4: File Handling and Class Implementation

a) Consider a class with fixed-length data members. Write a function to write to an opened binary file stream.

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

class Data {
    int id;
    double value;
public:
    void setData(int i, double v) { id = i; value = v; }
    void display() { cout << "ID: " << id << ", Value: " << value << endl; }
    void writeToFile(fstream &fs) {
        fs.seekp(0, ios::beg);
        fs.write(reinterpret_cast<char*>(this), sizeof(Data));
    }
};

int main() {
    fstream fs("data.dat", ios::binary | ios::in | ios::out | ios::trunc);
    Data data;
    data.setData(1, 99.9);
    data.writeToFile(fs);
    fs.close();
}
```

Question 5: Queue and Stack Implementation

a) Implement a Queue class using FIFO operations. Provide all required methods including push, pop, and peek.

```
template <typename T>
class Queue {
    T* arr;
    int front, rear, capacity;
public:
    Queue(int cap) : capacity(cap), front(0), rear(0) { arr = new T[capacity]; }
    void enqueue(T data) { if (rear == capacity) { cout << "Queue Full\n"; return; } arr[rear++] =
data; }
    T dequeue() { if (front == rear) { cout << "Queue Empty\n"; return T(); } return arr[front++];
}
    ~Queue() { delete[] arr; }
};

int main() {
    Queue<int> q(5);
    q.enqueue(10);
    q.enqueue(20);
    cout << q.dequeue() << endl;  // Output: 10
}
```