Question 1:

Write a main program that:

Dynamically creates a Player object and initializes it with user inputs. Outputs the current details of the player. Changes the player's details based on user input. Outputs the updated player details.

Input Format:

- Enter the player's name: player_name
- Enter the player's age: player_age
- Enter the game type: game_type
- Do you want to update player details? (y/n): choice
- (If choice is 'y') Enter the new player's name: new_player_name
- (If choice is 'y') Enter the new player's age: new_player_age
- (If choice is 'y') Enter the new game type: new_game_type

Output Format:

- Player's Name: player_name
- Player's Age: player_age
- Player's Game Type: game_type
- (If choice is 'y') Updated Player's Name: new_player_name
- (If choice is 'y') Updated Player's Age: new_player_age
- (If choice is 'y') Updated Player's Game Type: new_game_type

Title for Question 1: Details of the player-encapsulsation

Solution:

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

class Player {
private:
    string pName;
    int pAge;
    string gameType;
public:
    // Parameterized Constructor
    Player(string pName, int pAge, string gameType): pName(pName), pAge(p

    // Getter methods
    string getPName() {
        return pName;
    }
    int getPAge() {
```

```
        return pAge;
    }
    string getGameType() {
        return gameType;
    }

    // Method to change player details
    void changePlayerDetails(string newPName, int newPAge, string newGame
        pName = newPName;
        pAge = newPAge;
        gameType = newGameType;
    }
};

int main() {
    string name, gameType, newName, newGameType;
    int age, newAge;
    char choice;

    // Initial user input
    //cout << "Enter the player's name: ";
    cin >> name;
   // cout << "Enter the player's age: ";
    cin >> age;
  //  cout << "Enter the game type: ";
    cin >> gameType;

    // Create Player object
    Player ply(name, age, gameType);

    // Show current details
    cout << "Player's Name: " << ply.getPName() << endl;
    cout << "Player's Age: " << ply.getPAge() << endl;
    cout << "Player's Game Type: " << ply.getGameType() << endl;

    // Ask if the user wants to update details
   // cout << "Do you want to update player details? (y/n): ";
    cin >> choice;

    if (choice == 'y') {
      //  cout << "Enter the new player's name: ";
        cin >> newName;
       // cout << "Enter the new player's age: ";
        cin >> newAge;
       // cout << "Enter the new game type: ";
        cin >> newGameType;

        // Update Player details
        ply.changePlayerDetails(newName, newAge, newGameType);

        // Show updated details
        cout << "Updated Player's Name: " << ply.getPName() << endl;
        cout << "Updated Player's Age: " << ply.getPAge() << endl;
        cout << "Updated Player's Game Type: " << ply.getGameType() << en
    }

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	david 20 Basketball n	Player's Name: david Player's Age: 20 Player's Game Type: Basketball
2	Jack 33 Golf y Jill 34 Tennis	Player's Name: Jack Player's Age: 33 Player's Game Type: Golf Updated Player's Name: Jill Updated Player's Age: 34 Updated Player's Game Type: Tennis
3	Akil 25 Soccer y Bobin 24 Football	Player's Name: Akil Player's Age: 25 Player's Game Type: Soccer Updated Player's Name: Bobin Updated Player's Age: 24 Updated Player's Game Type: Football
4	Mary 28 Baseball n	Player's Name: Mary Player's Age: 28 Player's Game Type: Baseball
5	Paul 42 Chess y George 40 Poker	Player's Name: Paul Player's Age: 42 Player's Game Type: Chess Updated Player's Name: George Updated Player's Age: 40 Updated Player's Game Type: Poker
6	David 36 Hockey n	Player's Name: David Player's Age: 36 Player's Game Type: Hockey

White List:

Black List:

Question 2:

Write a simple parameterized constructor to change the input sentence to sentence case program. Get input from the user as a string and change the input to proper sentence case.

Input Format:

- The user will be prompted with the message: "Enter your sentence: ".
- The user should then input a sentence (or multiple sentences) in any case, which could be a mix of uppercase and lowercase letters. The sentence(s) may contain punctuation such as periods, exclamation points, and question marks.

Output Format:

- After inputting the sentence, the program will convert the sentence to sentence case.
- The converted sentence will be displayed with the message: "Converted sentence: " followed by the actual converted sentence.

Title for Question 2: Sentence case // parameterized constructor

Solution:

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

class SentenceCaseConverter {
private:
    string sentence;

public:
    // Parameterized constructor
    SentenceCaseConverter(string inputSentence) : sentence(inputSentence)
        convertToSentenceCase();
    }

    // Function to convert the sentence to sentence case
    void convertToSentenceCase() {
        bool newSentence = true;
        for (size_t i = 0; i < sentence.length(); ++i) {
            if (newSentence && isalpha(sentence[i])) {
                sentence[i] = toupper(sentence[i]);
                newSentence = false;
            } else if (!newSentence) {
                sentence[i] = tolower(sentence[i]);
            }

            if (sentence[i] == '.' || sentence[i] == '!' || sentence[i] =
                newSentence = true;
            }
        }
    }

    // Function to display the converted sentence
    void display() {
        cout << "Converted sentence: " << sentence << endl;
    }
};

int main() {
    string inputSentence;
    //cout << "Enter your sentence: ";
    getline(cin, inputSentence);

    SentenceCaseConverter converter(inputSentence);
    converter.display();

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	hELlo. how ARE YoU? GoOd.	Converted sentence: Hello. How are you? Good.
2	hello world how are you I am fine	Converted sentence: Hello world how are you i am fine
3	a.b.c!	Converted sentence: A.B.C!

S.No	Inputs	Outputs
4	##!!..%%	Converted sentence: ##!!..%%
5	hello world!how are you?I am fine	Converted sentence: Hello world!How are you?I am fine
6	42 is the answer to life, the universe, and everything	Converted sentence: 42 Is the answer to life, the universe, and everything

White List:

Black List:

Question 3:

The program should use a class called "Cube" with a constructor and destructor. The constructor should initialize the side length of the cube, and the destructor should display a message indicating that the cube has been destroyed. Additionally, the program should prompt the user to enter the side length of the cube and then display the calculated volume.

Input Format:

- The program should prompt the user with the message: "Enter the side of the cube: ".
- The user should then enter a numerical value representing the side length of the cube.

Output Format:

- Upon entering the side value, a message should be displayed saying that a cube with that side length has been created.
- The program should then display the calculated volume of the cube.
- As the program concludes, a message should be displayed saying the cube has been destroyed.

Title for Question 3: Volume of Cube//constructor and destructor

Solution:

```
#include <iostream>
class Cube {
private:
    double side;
public:
    // Constructor to initialize the side of the cube
    Cube(double s) : side(s) {
        std::cout << "Cube with side " << side << " is created." << std::
    }
    // Destructor
    ~Cube() {
        std::cout << "Cube with side " << side << " is destroyed." << std
    }

    // Function to calculate volume of the cube
```

```
    double volume() const {
        return side * side * side;
    }
};

int main() {
    double user_side;

    // Input format
   // std::cout << "Enter the side of the cube: ";
    std::cin >> user_side;

    // Create an object of the Cube class
    Cube myCube(user_side);

    // Output format
    std::cout << "Volume of cube with side " << user_side << " is: " << m

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	5	Cube with side 5 is created. Volume of cube with side 5 is: 125 Cube with side 5 is destroyed.
2	11.8	Cube with side 11.8 is created. Volume of cube with side 11.8 is: 1643.03 Cube with side 11.8 is destroyed.
3	0.08	Cube with side 0.08 is created. Volume of cube with side 0.08 is: 0.000512 Cube with side 0.08 is destroyed.
4	98	Cube with side 98 is created. Volume of cube with side 98 is: 941192 Cube with side 98 is destroyed.
5	-11	Cube with side -11 is created. Volume of cube with side -11 is: -1331 Cube with side -11 is destroyed.
6	5.56	Cube with side 5.56 is created. Volume of cube with side 5.56 is: 171.88 Cube with side 5.56 is destroyed.

White List:

Black List:

Question 4:

The program should use a class called "Factorial" with a copy constructor. The copy constructor should create a copy of the original object, and both the original and copied objects should display the computed factorial.

The program should use the copy constructor to create a copy of the original object and display the factorial for both objects.

Input Format:

The user is prompted to enter a non-negative integer as follows:

- Enter a number:

The program should then display the factorial of the entered number in the format:

- Factorial of [NUMBER] is: [RESULT]

[NUMBER] should be replaced by the integer number provided by the user.

Output Format:

[RESULT] should be replaced by the computed factorial of the number.

Title for Question 4: Factorial// copy constructor

Solution:

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

class Factorial {
private:
    int num;
    long long result;

public:
    // Constructor
    Factorial(int n) : num(n) {
        result = calculateFactorial(n);
    }

    // Copy constructor
    Factorial(const Factorial &f) {
        num = f.num;
        result = f.result;
    }

    // Recursive function to calculate factorial
    long long calculateFactorial(int n) {
        if (n == 0) {
            return 1;
        }
        return n * calculateFactorial(n - 1);
    }

    void display() {
        cout << "Factorial of " << num << " is: " << result << endl;
    }
};

int main() {
```

```
    int number;

    // i/p format
    //cout << "Enter a number: ";
    cin >> number;

    // Using constructor
    Factorial fact(number);

    // Using copy constructor
    Factorial copyFact(fact);

    // o/p format
    copyFact.display();

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	5	Factorial of 5 is: 120
2	9	Factorial of 9 is: 362880
3	10	Factorial of 10 is: 3628800
4	15	Factorial of 15 is: 1307674368000
5	33	Factorial of 33 is: 3400198294675128320
6	20	Factorial of 20 is: 2432902008176640000

White List:

Black List:

Question 5:

Create a program that simulates the construction of toy cars. The program should ask the user for the number of wheels, car bodies, and figures available. Then, the program should display how many complete toy cars can be made from these parts.

In addition, implement a class named ToyCar that should have a constructor to initialize the number of wheels, car bodies, and figures. The class should also have a member function called maxCars that returns the maximum number of complete toy cars that can be constructed.

The user will input three integers: w, b, and f representing the number of wheels, car bodies, and figures, respectively.

Input Format:

- The program expects three integers separated by spaces.

- These integers represent the number of wheels (w), car bodies (b), and figures (f), respectively.

Output Format:

- The program outputs a single line indicating the maximum number of complete toy cars that can be built.

Title for Question 5: Toy car workshop// constructor

Solution:

```
#include <iostream>

class ToyCar {
private:
    int wheels;
    int bodies;
    int figures;

public:
    ToyCar(int w, int b, int f) : wheels(w), bodies(b), figures(f) {}

    int maxCars() {
        int max_wheels = wheels / 4;
        int max_bodies = bodies;  // Since 1 body makes 1 car, we can jus
        int max_figures = figures / 2;

        int min_value = max_wheels;

        if (max_bodies < min_value) {
            min_value = max_bodies;
        }

        if (max_figures < min_value) {
            min_value = max_figures;
        }

        return min_value;
    }
};

int main() {
    int w, b, f;
    //std::cout << "Enter the number of wheels, bodies, and figures: ";
    std::cin >> w >> b >> f;

    ToyCar car(w, b, f);

    std::cout << "Max cars that can be built: " << car.maxCars() << std::

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	100 25 50	Max cars that can be built: 25
2	w b f	Max cars that can be built: 0
3	12 3 6	Max cars that can be built: 3
4	8 2 5	Max cars that can be built: 2
5	0 5 10	Max cars that can be built: 0
6	16 4 7	Max cars that can be built: 3

White List:

Black List:
