Answer Key

Part I: True/False (1 pt each)

1. True
2. True
3. False
4. False
5. True
6. True
7. False
8. False
9. False

Part II: Multiple choice (1.5 pts each)

1. E
2. A
3. B
4. C
5. D
6. E
7. F
8. E
9. A
10. A
11. B
12. C
13. D
14. E

Part III: Debug and predict the output

1) 2 Pts

Error:

- line 3 & 6, improper assignment of pointer

Output:

- 30
- 11
- Garbage data

2) 2.5 Pts

Error:

- Line 3 - header file missed
- line 7 - improper insertion of vector element

Output:

- Vector size and capacity: 24
- First and last element: 2013

3) 2 Pts

- line 7 & 8 - member access operator (.) cannot be used with structure pointer

Output:

- Result = 9/10

4) 2.5 Pts

- line 3 - header file missed
- line 6 - unique pointer does not allow assignment and copy operation
- line 9 - unique pointer content doesn't print like raw pointer

Output:

- Address pointed by ptr1
- 10A
- 2

Part IV: Write a program

1) – 2 Pts

```
int product[][5] =  {  16, 22, 99, 4, 18,
                       -78, 4, 11, 5, 98,
                       55, 6, 15, 2, 45,
                       33, 88, 72, 16, 3
                    };

int* ptr = &product[0][0];

for (int i = 0; i < 20; i++) {
    cout << *(ptr + i) << " ";

    if ((i + 1) % 5 == 0)
        cout << endl;
}
```

2) – 5 Pts

```
#include <iostream>
#include <vector>
#include <cmath>
#include <iomanip>

using namespace std;

// Template function
template <typename T>
void statistics(vector<T>& dataSet, int size) {
    T value;
    double mean = 0.0;
    double variance = 0.0;
    double stdDeviation = 0.0;

    // Read data set
    cout << "\n Enter " << size << " data values:\n";
    for (int i = 0; i < size; i++)   {
        cin >> value;
        dataSet.push_back(value);
    }

    // Compute means
    double sum = 0.0;
    for (int i = 0; i < size; i++)  {
        sum += dataSet[i];
    }

    mean = sum / size;
```

```
    // Compute variance
    for (int i = 0; i < size; i++)    {
        variance += pow(dataSet[i] - mean, 2);
    }

    variance /= size;

    // Compute standard deviation
    stdDeviation = sqrt(variance);

    // Display data set
    cout << "\nData Set:\n";
    for (const auto& element : dataSet)  {
        cout << element << " ";
    }

    // Display results
    cout << fixed << setprecision(4);
    cout << "\n\nMean (μ)              = " << mean;
    cout << "\nVariance (σ²)         = " << variance;
    cout << "\nStandard Deviation(S) = " << stdDeviation << endl;
}

int main() {
    int size, choice;

    cout << "Enter size of the data set: ";
    cin >> size;
    cout << "\nSelect Data Type\n";
    cout << "1. short int\n"<<"2. long int\n" << "3. float\n";
    cout << "4. double\n" << "Enter choice: ";
    cin >> choice;
```

```
    switch (choice) {
        case 1:
            vector<short> data;
            statistics(data, size);
            break;

        case 2:
            vector<long> data;
            statistics(data, size);
            break;

        case 3:
            vector<float> data;
            statistics(data, size);
            break;

        case 4:
            vector<double> data;
            statistics(data, size);
            break;

        default:
            cout << "Invalid choice!" << endl;
    }

    return 0;
}
```

3) – 4 Pts

```
#include <iostream>
#include <fstream>
#include <vector>
#include <iomanip>

using namespace std;

struct Vehicle {
    int code;
    string model;
    string color;
    int year;
    double price;
};

int main() {
    vector<Vehicle> vehicles;
    Vehicle v;

    // Open file for reading
    ifstream fin("vehicle.txt");
    if (!fin) {
        cout << "Error: Unable to open vehicle.txt for reading." << endl;
        return 1;
    }

    // Read records
    while (fin >> v.code >> v.model >> v.color>> v.year >> v.price) {
        vehicles.push_back(v);
    }

    fin.close();
```

```
int searchCode;
bool found = false;

cout << "Enter Vehicle Code to update: ";
cin >> searchCode;

for (auto &vehicle : vehicles) {
    if (vehicle.code == searchCode) {
        found = true;

        cout << "\nVehicle Found\n";
        cout << "Code  : " << vehicle.code << endl;
        cout << "Model : " << vehicle.model << endl;
        cout << "Color : " << vehicle.color << endl;
        cout << "Year  : " << vehicle.year << endl;
        cout << "Price : $" << fixed
            << setprecision(2)<< vehicle.price << endl;

        char choice;
        cout << "\nUpdate Model? (Y/N): ";
        cin >> choice;
        if (toupper(choice) == 'Y')  {
            cout << "Enter New Model: ";
            cin >> vehicle.model;
        }

        cout << "Update Color? (Y/N): ";
        cin >> choice;
        if (toupper(choice) == 'Y')  {
            cout << "Enter New Color: ";
            cin >> vehicle.color;
        }
```

```
        cout << "Update Year? (Y/N): ";
        cin >> choice;
        if (toupper(choice) == 'Y')  {
            cout << "Enter New Year: ";
            cin >> vehicle.year;
        }

        cout << "Update Price? (Y/N): ";
        cin >> choice;
        if (toupper(choice) == 'Y') {
            cout << "Enter New Price: ";
            cin >> vehicle.price;
        }

        break;
    }
}

if (!found)   {
    cout << "\nVehicle Code not found." << endl;
    return 0;
}

// Open file for writing
ofstream fout("vehicle.txt");

if (!fout)  {
    cout << "Error: Unable to open vehicle.txt for writing." << endl;
    return 1;
}
```

```
    // Write updated records back to file
    for (const auto &vehicle : vehicles) {
        fout << vehicle.code << " " << vehicle.model << " "
             << vehicle.color << " "<< vehicle.year << " "
             << fixed << setprecision(2)<< vehicle.price << endl;
    }

    fout.close();

    cout << "\nVehicle record updated successfully." << endl;

    return 0;
}
```

Evaluation Rubrics

1. Objective Item Rubrics (True/False and Multiple Choice)

Correct Answer (full mark)	Wrong Answer (zero mark)
Correctly identifies the statement as TRUE or FALSE.	incorrectly identifies the statement as TRUE or FALSE..

2. Multiple Choice Rubrics

Complete Answer (full mark)	Wrong Answer (zero mark)
Choose the correct answer from the available options.	Choose an incorrect answer form the provide choice.

3. Debugging and Output Prediction Rubrics

Proficient (full mark)	Competent (50% – 75%)	Satisfactory (25% - 50%)	Poor (zero mark)
Correctly debug and predict the output and print it as expected	Correctly debug and predict the output with minor issue or the output is not printed as expected.	Partially correct debugging and predictions, missing key details.	Unable to debug and predict the output correctly.

4. Program writing rubrics

Evaluation Rubric for Q #1:

Criterion	Correct use of a pointer to access array elements	Correct use of a single loop to print all 20 elements	**Total**
Marks	**1**	**1**	**2 marks**

Evaluation Rubrics

Evaluation Rubric for Q #2: Generic Statistics Calculator Using Function Templates

Criterion	Template Function Implementation	Vector and Data Storage	Statistical Computation	Menu and Data Type Selection	Input/Output and Result Presentation	**Total**
Marks	**1**	**1**	**1**	**1**	**1**	**5 marks**

Criteria	**Proficient (5)**	**Competent (4)**	**Satisfactory (3)**	**Limited (2)**	**Poor (1)**
Template Function Implementation	Template function is correctly implemented and supports all specified data types effectively.	Template function works correctly with minor issues.	Template implementation is partially correct.	Significant errors in template implementation.	Template concept is not implemented correctly.
Vector and Data Storage	Data is correctly stored using vectors and managed efficiently.	Vector usage is mostly correct with minor issues.	Basic vector operations are implemented but contain errors.	Vector usage is poorly implemented.	Vector is not used correctly.
Statistical Computation (Mean, Variance, Standard Deviation)	All statistical calculations are accurate and implemented correctly.	Calculations are mostly accurate with minor errors.	Some calculations are correct, but noticeable errors exist.	Significant errors in calculations.	Calculations are incorrect or missing.
Menu and Data Type Selection	Menu is fully functional and correctly handles all data type selections.	Menu functions correctly with minor issues.	Some menu options work correctly.	Menu implementation is incomplete or inconsistent.	Menu functionality is missing or incorrect.
Input/Output and Result Presentation	User prompts are clear; results are accurately displayed and properly formatted.	Input/output handling is mostly effective with minor formatting issues.	Output formatting or prompts need improvement.	User interaction is confusing or poorly formatted.	Input/output handling is largely incorrect.

Evaluation Rubrics

Evaluation Rubric for Program Q #3 - Vehicle Record Management using File Handling + Vector + Structure

Criterion	File Handling	Structure and Data Management	Search and Update Functionality	Input/Output and Program Logic	**Total**
Marks	**1**	**1**	**1**	**1**	**4 marks**

Criteria	**Proficient (4)**	**Competent (3)**	**Satisfactory (2)**	**Limited (1)**
File Handling (Reading & Writing)	Correctly opens, reads, updates, and writes vehicle records. Proper error handling implemented.	File operations mostly correct with minor errors. Error handling partially implemented.	Basic file operations work but contain several errors.	File handling is incorrect or incomplete.
Structure and Data Management	Appropriate use of structure (Vehicle) and vector for record management. Data is stored and manipulated correctly.	Structure and vector are used adequately with minor issues.	Partial implementation of structure/vector usage.	Structure/vector usage is incorrect or missing.
Search and Update Functionality	Vehicle search and selective field updates are fully implemented and function correctly.	Search and update operations work with minor issues.	Search or update functionality is partially implemented.	Search and update functionality is incorrect or missing.
Input/Output and Program Logic	Clear prompts, formatted display, and logical program flow. User interaction is professional and error-free.	User interaction is generally clear with minor formatting or logic issues.	Some prompts or outputs are unclear; logic requires improvement.	Poor user interaction and program flow.