**061306T4CSC**

**COMPUTER SCIENCE LEVEL 6**

**ICT/OS/CS/CR/08/6/A**

**Understand Algorithms and Data Structures**

**March/April 2025**

**TVET CURRICULUM DEVELOPMENT, ASSESSMENT AND CERTIFICATION COUNCIL (TVET CDACC)**



**PRACTICAL ASSESSMENT**

**INSTRUCTIONS TO ASSESSOR**

1. Assess the candidate as the practical progresses observing the critical areas
2. You are required to mark the practical as the candidate perform the tasks
3. You are required to take video clips at critical points
4. Ensure the candidate has a name tag and registration code at the back and front

## OBSERVATION CHECKLIST

| | | | |
|---|---|---|---|
| **Candidate's name & Registration No.** | | | |
| **Assessor's name & Id code** | | | |
| **Unit(s) of Competency** | **Understand Algorithms and Data Structures** | | |
| **Venue of Assessment** | | | |
| **Date of assessment** | | | |
| **Items to be evaluated:** *Please award marks as appropriate. Give a brief comment on your observation.* | **Marks allocated** | **Marks obtained** | **Comments** |
| **TASK 1: Demonstrate Array Operations** | | | |
| 1. Observed computer laboratory rules *(Award 2 marks or zero)* | 2 | | |
| 2. Launched the C++ IDE correctly *(Award 1 mark or zero)* | 1 | | |
| 3. Designed a user interface with the following features<br>- *Add student records*<br>- *Delete Student records*<br>- *Update student records*<br>- *View Student records*<br>*(Award 2 marks for each feature or zero)* | 8 | | |
| 4. Declared a class correctly<br>- *declaration*<br>*(Award 2 marks or zero)* | 2 | | |
| 5. Initialized an array to store 30 students<br>- *array*<br>*(Award 1 mark or zero)* | 1 | | |
| 6. Implemented a function to store student details.<br>- *Adm No*<br>- *Gender*<br>- *Name*<br>*(Award 2 marks for the function name, 1 mark each for the student details or zero)* | **5** | | |
| 7. Implemented a function to delete students<br>- *Adm No 003*<br>- *Adm No 005*<br>- *Adm No 004* | **5** | | |

| | | | |
|---|---|---|---|
| *(Award 2 marks for the function name, 1 mark each for deleted student details or zero)* | | | |
| 8. Implemented a function to update array<br>- *Student 1*<br>- *Student 2*<br>*(Award 2 marks for the function name, 1 mark each for the updated student details or zero)* | 4 | | |
| 9. Implemented a function to display current record of students.<br>- *displayed records*<br>- *displayed in correct order*<br>*(Award 2 marks for displaying records, 1 mark for displaying in correct order or zero)* | 3 | | |
| 10. Invoked the functions in the main program<br>- *Function to add student details*<br>- *Function to delete student details*<br>- *Function to update student details*<br>- *Function to view student details*<br>*(Award 1 mark or zero)* | 4 | | |
| **Sub-Total 1** | **35** | | |
| **TASK 2: Implement Linear Search.** | | | |
| 11. Added a feature to search on the user interface<br>*(Award 2 marks or zero)* | **2** | | |
| 12. Implemented linear search algorithm to search for the student<br>*-Adm 002*<br>*(Award 3 marks or zero)* | **3** | | |
| 13. Displayed an appropriate message<br>- *The record was not found*<br>*(Award 2 marks or zero)* | **2** | | |
| 14. Implemented linear search algorithm to search for the student<br>-Adm 001<br>*(Award 3 marks or zero)* | **3** | | |
| 15. Displayed an appropriate message<br>- *The record was found*<br>*(Award 2 marks or zero)* | **2** | | |
| 16. Program executed correctly<br>*(Award 3 marks or zero)* | **3** | | |

| | | | |
|---|---|---|---|
| **Sub- Total 2** | **15** | | |
| **GRAND TOTAL** | **50** | | |

**ASSESSMENT OUTCOME**

The candidate was found to be:

Competent ☐ Not yet Competent ☐

*(Please tick as appropriate)*

*(The candidate is competent if the candidate obtains at least 50%)*

**Feedback from the Candidate:**

**Feedback to the Candidate:**



**Sample code for Task 1 and 2**

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

class Student {
public:
    string admNo;
    string name;
    string gender;

    // Constructor
    Student(string admNo = "", string name = "", string gender = "")
        : admNo(admNo), name(name), gender(gender) {}

    // Method to display student details
```

```
    void display() {
        cout << "Adm No: " << admNo << ", Name: " << name << ", Gender: " << gender << endl;
    }
};

class ClassRegister {
private:
    Student students[30];
    int totalStudents;

public:
    // Constructor to initialize the total students to 0
    ClassRegister() : totalStudents(0) {}

    // Function to add students at a specific position
    void addStudent(int index, string admNo, string name, string gender) {
        if (index >= 0 && index < 30 && totalStudents < 30) {
            students[index] = Student(admNo, name, gender);
            totalStudents++;
        } else {
            cout << "Invalid index or class is full." << endl;
        }
    }

    // Function to delete a student by Adm No
    void deleteStudent(string admNo) {
        bool found = false;
        for (int i = 0; i < totalStudents; i++) {
            if (students[i].admNo == admNo) {
                // Shift all subsequent students down to fill the gap
                for (int j = i; j < totalStudents - 1; j++) {
```

```
            students[j] = students[j + 1];
        }
        totalStudents--;
        found = true;
        cout << "Student with Adm No " << admNo << " deleted." << endl;
        break;
    }
  }
  if (!found) {
    cout << "Student not found." << endl;
  }
}

// Function to update a student's details
void updateStudent(string admNo, string newName, string newGender) {
  bool found = false;
  for (int i = 0; i < totalStudents; i++) {
    if (students[i].admNo == admNo) {
      students[i].name = newName;
      students[i].gender = newGender;
      found = true;
      cout << "Student details updated." << endl;
      break;
    }
  }
  if (!found) {
    cout << "Student not found." << endl;
  }
}

// Function to display all student records from last to first
```

```
    void displayAllStudents() {
        cout << "Displaying student records from last to first:" << endl;
        for (int i = totalStudents - 1; i >= 0; i--) {
            students[i].display();
        }
    }

    // Function to search for a student by Adm No or Name using linear search
    void searchStudent(string searchTerm) {
        bool found = false;
        for (int i = 0; i < totalStudents; i++) {
            if (students[i].admNo == searchTerm || students[i].name == searchTerm) {
                students[i].display();
                found = true;
                break;
            }
        }
        if (!found) {
            cout << "Student not found." << endl;
        }
    }
};

int main() {
    ClassRegister register1;

    // Step 4: Storing the initial records
    register1.addStudent(0, "001", "John Doe", "Male");
    register1.addStudent(1, "002", "Jane Smith", "Female");
    register1.addStudent(2, "003", "Robert Brown", "Male");
    register1.addStudent(3, "004", "Emily Davis", "Female");
```

```
    register1.addStudent(4, "005", "Michael Johnson", "Male");
    register1.addStudent(5, "006", "Sarah Wilson", "Female");

    // Step 5: Deleting students with Adm No 003, 005, 004
    register1.deleteStudent("003");
    register1.deleteStudent("005");
    register1.deleteStudent("004");

    // Step 6: Updating the records of two new students
    register1.updateStudent("001", "John Doe Updated", "Male");
    register1.updateStudent("002", "Jane Smith Updated", "Female");

    // Step 7: Displaying current records from last to first
    register1.displayAllStudents();

    // Step 8: Searching for a student by Adm No or Name
    string searchTerm;
    cout << "\nEnter student Adm No or Name to search: ";
    cin >> searchTerm;
    register1.searchStudent(searchTerm);

    return 0;
}
```