NAME GIKURU JOSEPH NDERITU

REG: SCT 212-0574/2022

COURSE: BSc COMPUTER TECNOLOGY

YEAR:2.2

ASSIGNMENT: DATA STRUCTURES AND ALGORITHM LAB 1

Task One (1)

Here is a step-by-step solution to implement the two functions and the main function in C++:

1. Define the summation function. This function takes an array and its length as parameters. It initializes a variable sum to 0, then iterates over the array, adding each element to sum. Finally, it returns sum.

```cpp int summation (int arr[],
int n) {  int sum = 0;  for(int i
= 0; i < n; i++) {  sum +=
arr[i];
  }
  return sum;
}
```

2. Define the maximum function. This function also takes an array and its length as parameters. It initializes a variable max to the first element of the array, then iterates over the array. If it finds an element greater than max, it updates max. Finally, it returns max.

```cpp int maximum(int arr[],
int n) {  int max = arr[0];
```

```
   for(int i = 1; i < n; i++) {
if(arr[i] > max) {
max = arr[i];
      }
   }
   return max;
}
```

3. In the main function, declare an array of length n. Ask the user for the value of n, then allow the user to enter these n integers, storing them in the array.

```cpp int
main() {
   int n;
   cout << "Enter the number of elements: ";
   cin >> n;
int arr[n];
   cout << "Enter the elements: ";
   for(int i = 0; i < n; i++) {
cin >> arr[i];
   }
```

4. Call the two functions in turns and display their outputs.

```cpp    cout << "Sum: " << summation(arr, n)
<< endl;    cout << "Max: " << maximum(arr, n)
<< endl;    return 0;
}
```

Task Two (2)

Here is a step-by-step solution to implement the system in C++ using arrays and structs:

1. Define the struct for Grade. This struct has two members: Mark and the_grade. The_grade is a char that is calculated from Mark using a grading system.

```cpp
struct Grade {
    int Mark;
char the_grade;
};
```

2. Define the struct for Course. This struct has two members: course_code and Course_name, both of which are strings.

```cpp
struct Course {
string course_code;
string Course_name;
};
```

3. Define the struct for Student. This struct has five members: registration number, name, age, course, and grades. The last two are objects of the structs defined above.

```cpp
struct Student {
string registration_number;
string name;
    int age;
    Course course;
```

```
    Grade grades;
};
```

4. In the main function, declare an array of Students of length 40. This will allow you to add at most 40 students.

```cpp int
main() {
    Student students[40];
```

5. To add a student, ask the user for the student's details, then store them in the next available spot in the array.

```cpp     int num_students = 0;     cout << "Enter student's registration number, name, age, course
code, course name, and mark: ";
    cin >> students[num_students].registration_number >> students[num_students].name >>
students[num_students].age >> students[num_students].course.course_code >>
students[num_students].course.Course_name >> students[num_students].grades.Mark;
    students[num_students].grades.the_grade =
calculate_grade(students[num_students].grades.Mark);
num_students++;
```

6. To edit a student's details, ask the user for the student's registration number, then find the student in the array and update their details.

```cpp
    string reg_num;     cout << "Enter registration number
of student to edit: ";     cin >> reg_num;     for(int i = 0; i <
num_students; i++) {
```

```
if(students[i].registration_number == reg_num) {
cout << "Enter new details: ";
        cin >> students[i].name >> students[i].age >> students[i].course.course_code >>
students[i].course.Course_name >> students[i].grades.Mark;
students[i].grades.the_grade = calculate_grade(students[i].grades.Mark);
        break;
      }
   }
```

7. To add marks and calculate grades, you can use a function calculate grade that takes a mark as input and returns the corresponding grade.

```cpp char calculate grade(int
mark) {     if(mark > 69) {
return 'A';     } else if(mark >
59) {
      return 'B';     }
else if(mark > 49) {
return 'C';     } else
if(mark > 39) {
return 'D';     } else {
return 'E';
   }
}
```

8. To ensure the grades, once calculated, cannot be altered, you can make the grade a const member of the Grade struct. This means that once it is set, it cannot be changed.

```cpp struct Grade {

int Mark;  const char

the grade;

};

```

Task Three (3)

1. Define the class for Grade. This class has two private members: Mark and the grade. The grade is a char that is calculated from Mark using a grading system. The class has public methods to get and set Mark, and to get the grade.

```cpp class

Grade {

private:

  int Mark;

char the_grade;

public:

  void setMark(int mark) {

Mark = mark;

calculate_grade();

  }

  int getMark() {

return Mark;

  }

  char getGrade() {

return the_grade;

  }

  void calculate_grade() {

    if(Mark > 69) {

the_grade = 'A';      }

else if(Mark > 59) {
```

```
the_grade = 'B';        }
else if(Mark > 49) {
the_grade = 'C';        }
else if(Mark > 39) {
the_grade = 'D';
      } else {
         the_grade = 'E';
      }
   }
};
```

2. Define the class for the Course. This class has two private members: course code and Course name, both of which are strings. The class has public methods to get and set these members.

```cpp
class
Course {
private:
   string course_code;
string Course_name; public:
   void setCourseCode(string code) {
course_code = code;
   }
   string getCourseCode() {
return course_code;
   }
   void setCourseName(string name) {
      Course_name = name;
   }
   string getCourseName() {
return Course_name;
```

```
    }
};
```

3. Define the class for Students. This class has five private members: registration number, name, age, course, and grades. The last two are objects of the classes defined above. The class has public methods to get and set these members.

```cpp
class Student {
private:
    string registration number;
string name;
    int age;
    Course course;
Grade      grades;
public:
    void setRegistrationNumber(string reg_num) {
registration_number = reg_num;
    }
    string getRegistrationNumber() {
return registration_number;
    }
    void setName(string n) {
name = n;
    }
    string getName() {
        return name;
    }
    void setAge(int a) {
        age = a;
    }
```

```
   int getAge() {
return age;
   }
   void setCourse(Course c) {
course = c;
   }
   Course getCourse() {
return course;
   }
   void setGrades(Grade g) {
      grades = g;
   }
   Grade getGrades() {
return grades;
   }
};
```

Task Four (4)

The UML diagram for the ADT List would include a box labelled "List" with two compartments. The first compartment would list the data members of the List, which could include an array to hold the list elements and an integer to keep track of the size of the list. The second compartment would list the operations on the List, which could include methods to add an element to the list, remove an element from the list, get the size of the list, and check if the list is empty.

- insert(item, position): Inserts an item at a given position in the list.

- delete(position): Removes the item at a given position.

- retrieve(position): Returns the item at a given position.

- length(): Returns the number of items in the list.

- isEmpty(): Returns true if the list is empty, false otherwise.

The diagram would also show that the List has a private attribute, an array to hold the list items.