NAME: BONIFACE MWANGI WARIGI

REG NO: SCT212-0726/2022

Data Structures and Algorithms: LAB ONE

Task One (1)

Here is a step-by-step solution to the problem:

1. First, we need to define the two functions, summation and maximum. Both functions will take an array of integers as input.

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

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;
}
```

2. In the main function, we will declare an array of length n, get the value of n from the user, and then allow the user to enter these n integers, storing them in the array.

```c++
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];
   }
   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 the problem:

1. First, we need to define the structures for Course, Grade, and Student.

```c++
struct Course {
   string course_code;
   string course_name;
};

struct Grade {
   int mark;
   char grade;
};

struct Student {
   string reg_no;
   string name;
```

```
    int age;
    Course course;
    Grade grade;
};
```

2. We can then create an array of Students and provide functions to add a student, edit a student's details, and add marks and calculate grades.

```c++
Student students[40];
int student_count = 0;

void addStudent(Student student) {
    if(student_count < 40) {
        students[student_count++] = student;
    }
}

void editStudent(int index, Student student) {
    if(index >= 0 && index < student_count) {
        students[index] = student;
    }
}

void addMarks(int index, int mark) {
    if(index >= 0 && index < student_count) {
        students[index].grade.mark = mark;
        if(mark > 69) {
            students[index].grade.grade = 'A';
        } else if(mark > 59) {
            students[index].grade.grade = 'B';
        } else if(mark > 49) {
            students[index].grade.grade = 'C';
        } else if(mark > 39) {
            students[index].grade.grade = 'D';
```

```
        } else {
            students[index].grade.grade = 'E';
        }
    }
}
```

Task Three (3)

Here is a step-by-step solution to the problem:

1. First, we need to define the classes for Course, Grade, and Student.

```c++
class Course {
public:
    string course_code;
    string course_name;
};

class Grade {
public:
    int mark;
    char grade;
};

class Student {
public:
    string reg_no;
    string name;
    int age;
    Course course;
    Grade grade;
};
```

2. We can then create an array of Students and provide functions to add a student, edit a student's details, and add marks and calculate grades.

```c++
Student students[40];
int student_count = 0;

void addStudent(Student student) {
   if(student_count < 40) {
      students[student_count++] = student;
   }
}

void editStudent(int index, Student student) {
   if(index >= 0 && index < student_count) {
      students[index] = student;
   }
}

void addMarks(int index, int mark) {
   if(index >= 0 && index < student_count) {
      students[index].grade.mark = mark;
      if(mark > 69) {
         students[index].grade.grade = 'A';
      } else if(mark > 59) {
         students[index].grade.grade = 'B';
      } else if(mark > 49) {
         students[index].grade.grade = 'C';
      } else if(mark > 39) {
         students[index].grade.grade = 'D';
      } else {
         students[index].grade.grade = 'E';
      }
   }
}
```

Task Four (4)

The UML diagram for the ADT List would have a box labeled "List" with the following operations:

- 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.