**NAME** : FELIX OUMA

**REG NO :** SCT212-0169/2022

**UNIT :** DATA STRUCTURES AND ALGORITHMS

**Assignment lab 1**

Define two functions: summation and maximum both of which take an integer array of length

. The summation function gets the sum of the integers while the maximum function obtains the largest integer in the array.

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;

}

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.

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**

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

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 grades;

};

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

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 addMark(int index, int mark) {

if(index >= 0 && index < student\_count) {

students[index].grades.mark = mark;

if(mark > 69) {

students[index].grades.grade = 'A';

} else if(mark > 59) {

students[index].grades.grade = 'B';

} else if(mark > 49) {

students[index].grades.grade = 'C';

} else if(mark > 39) {

students[index].grades.grade = 'D';

} else {

students[index].grades.grade = 'E';

}

}

}

**TASK THREE**

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

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 grades;

};

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

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 addMark(int index, int mark) {

if(index >= 0 && index < student\_count) {

students[index].grades.mark = mark;

if(mark > 69) {

students[index].grades.grade = 'A';

} else if(mark > 59) {

students[index].grades.grade = 'B';

} else if(mark > 49) {

students[index].grades.grade = 'C';

} else if(mark > 39) {

students[index].grades.grade = 'D';

} else {

students[index].grades.grade = 'E';

}

}

}

**TASK FOUR**

Cary out a specification of ADT List using UML diagrams.

The UML diagram for the ADT List would include a class named List with the following operations:

* insert: This operation adds an element to the list at a specified position.
* delete: This operation removes an element from the list at a specified position.
* find: This operation returns the position of a specified element in the list.
* retrieve: This operation returns the element at a specified position in the list.
* size: This operation returns the number of elements in the list.
* isEmpty: This operation checks if the list is empty.
* isFull: This operation checks if the list is full.

The UML diagram would show the List class with these operations, and arrows indicating the direction of operation (from the user to the List class). The operations would be listed in a box under the class name, with the input and output types specified.