**DATA STRUCTURES AND ALGORITHM**

**NJENGA BEATRICE WAMBUI**

**SCT212-0153/2022**

**Task One (1)**

**Define two functions: summation and maximum both of which take an integer array of length 0<= n<= ∞. The summation function gets the sum of the integers while the maximum function obtains the largest integer in the array. Within the main function; declare an array of length n, obtain the n from the user, then allow the user to enter these n integers storing them in the array. Call the two functions in turns and display their outputs.**

def summation(arr):

return sum(arr)

def maximum(arr):

return max(arr)

def main():

try:

n = int(input("Enter the length of the array (n): "))

if n < 0:

print("Please enter a non-negative integer for the array length.")

return

user\_array = []

for i in range(n):

element = int(input(f"Enter integer {i+1}/{n}: "))

user\_array.append(element)

# Call the summation function

sum\_result = summation(user\_array)

print(f"Sum of the array: {sum\_result}")

# Call the maximum function

max\_result = maximum(user\_array)

print(f"Maximum of the array: {max\_result}")

except ValueError:

print("Please enter valid integers.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

PS C:\Users\hp> & C:/Users/hp/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/hp/Documents/lab.py

Enter the length of the array (n): 2

Enter integer 1/2: 1

Enter integer 2/2: 6

Sum of the array: 7

Maximum of the array: 6

**Task two (2)**

**Given that a student has the following details:**

**i. registration number (a string of at most 20 characters)**

**ii. name (A string of at most 50 characters)**

**iii. age (and integer)**

**iv. course (An object of a defined type course)**

**v. grades (An object of a defined type Grade)**

**A course has two major attributes namely:**

**i) course\_code: a string.**

**ii) Course\_name: a string**

**A grade has the following components:**

**i) Mark: an integer between 0 and 100.**

**ii) the\_grade: a character obtained from the mark through a grading system (if mark>69**

**the\_grade is A, if mark>59 but less than 70, the grade is B, if mark>49 but less than 60**

**the grade is C, if mark>39 but less than 50 then grade is D otherwise grade is E )**

**Using arrays and structures (structs), design this system and implement in C/C++ and be able to:**

**i) Add at most 40 students.**

**ii) Edit a student’s details**

**iii) Add marks and calculate grades.**

**iv) Ensure the grades, ones calculated, can not be altered.**

#include <iostream>

#include <string>

// Define the Grade structure

struct Grade {

int mark;

char the\_grade;

};

// Define the Course structure

struct Course {

std::string course\_code;

std::string course\_name;

};

// Define the Student structure

struct Student {

std::string reg\_number;

std::string name;

int age;

Course course;

Grade grade; // Note: This holds the calculated grade, and it cannot be altered after calculation

};

// Function to calculate grade based on the mark

char calculateGrade(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';

}

// Function to add a student

void addStudent(Student students[], int& numStudents, int maxStudents) {

if (numStudents < maxStudents) {

Student newStudent;

std::cout << "Enter Registration Number: ";

std::cin >> newStudent.reg\_number;

std::cout << "Enter Name: ";

std::cin.ignore(); // Clear the newline character left in the buffer

std::getline(std::cin, newStudent.name);

std::cout << "Enter Age: ";

std::cin >> newStudent.age;

std::cout << "Enter Course Code: ";

std::cin >> newStudent.course.course\_code;

std::cout << "Enter Course Name: ";

std::cin.ignore();

std::getline(std::cin, newStudent.course.course\_name);

// Initialize grade to an invalid mark; it will be calculated later

newStudent.grade.mark = -1;

newStudent.grade.the\_grade = ' ';

students[numStudents++] = newStudent;

std::cout << "Student added successfully." << std::endl;

} else {

std::cout << "Cannot add more students. Maximum limit reached." << std::endl;

}

}

// Function to edit a student's details

void editStudent(Student students[], int numStudents) {

std::string regNumber;

std::cout << "Enter Registration Number of the student to edit: ";

std::cin >> regNumber;

for (int i = 0; i < numStudents; ++i) {

if (students[i].reg\_number == regNumber) {

// Allow editing of some details

std::cout << "Enter new Age: ";

std::cin >> students[i].age;

std::cout << "Student details edited successfully." << std::endl;

return;

}

}

std::cout << "Student with Registration Number " << regNumber << " not found." << std::endl;

}

// Function to add marks and calculate grades for a student

void addMarksAndCalculateGrades(Student students[], int numStudents) {

std::string regNumber;

std::cout << "Enter Registration Number of the student to add marks: ";

std::cin >> regNumber;

for (int i = 0; i < numStudents; ++i) {

if (students[i].reg\_number == regNumber) {

// Allow adding marks and calculate grades only if not done before

if (students[i].grade.mark == -1) {

std::cout << "Enter Marks for " << students[i].name << ": ";

std::cin >> students[i].grade.mark;

// Calculate grade based on the mark

students[i].grade.the\_grade = calculateGrade(students[i].grade.mark);

std::cout << "Marks and Grades added successfully." << std::endl;

} else {

std::cout << "Marks and Grades already added for this student." << std::endl;

}

return;

}

}

std::cout << "Student with Registration Number " << regNumber << " not found." << std::endl;

}

// Function to display student details

void displayStudentDetails(const Student& student) {

std::cout << "Registration Number: " << student.reg\_number << std::endl;

std::cout << "Name: " << student.name << std::endl;

std::cout << "Age: " << student.age << std::endl;

std::cout << "Course Code: " << student.course.course\_code << std::endl;

std::cout << "Course Name: " << student.course.course\_name << std::endl;

std::cout << "Grade: " << student.grade.the\_grade << std::endl;

std::cout << std::endl;

}

// Function to display all students

void displayAllStudents(const Student students[], int numStudents) {

for (int i = 0; i < numStudents; ++i) {

std::cout << "Student #" << i + 1 << std::endl;

displayStudentDetails(students[i]);

}

}

int main() {

const int maxStudents = 40;

Student students[maxStudents];

int numStudents = 0;

int choice;

do {

std::cout << "Menu:" << std::endl;

std::cout << "1. Add Student" << std::endl;

std::cout << "2. Edit Student Details" << std::endl;

std::cout << "3. Add Marks and Calculate Grades" << std::endl;

std::cout << "4. Display All Students" << std::endl;

std::cout << "5. Exit" << std::endl;

std::cout << "Enter your choice: ";

std::cin >> choice;

switch (choice) {

case 1:

addStudent(students, numStudents, maxStudents);

break;

case 2:

editStudent(students, numStudents);

break;

case 3:

addMarksAndCalculateGrades(students, numStudents);

break;

case 4:

displayAllStudents(students, numStudents);

break;

case 5:

std::cout << "Exiting the program." << std::endl;

break;

default:

std::cout << "Invalid choice. Please enter a valid option." << std::endl;

}

} while (choice != 5);

return 0;

}

**Task 3**

**Use C++ classes to represent the scenario above**

#include <iostream>

#include <string>

class Student {

private:

std::string regno;

std::string name;

int age;

std::string registered[5];

std::string course;

std::string course\_id;

int unitmarks[5];

public:

void getData() {

std::cout << "Reg no: ";

std::cin >> regno;

std::cout << "Student name: ";

std::cin >> name;

std::cout << "Student age: ";

std::cin >> age;

std::cout << "Course: ";

std::cin >> course;

std::cout << "Course id: ";

std::cin >> course\_id;

std::cout << "Number of courses: ";

int num\_Course;

std::cin >> num\_Course;

for (int i = 0; i < num\_Course; i++) {

std::cout << "Enter unit " << i + 1 << " name: ";

std::cin >> registered[i];

std::cout << "Enter mark " << i + 1 << ": ";

std::cin >> unitmarks[i];

}

}

float calculateAverage() {

int sum = 0;

float count = 0;

for (int i = 0; i < 5; i++) {

if (!registered[i].empty()) {

sum += unitmarks[i];

count++;

}

}

if (count > 0) {

return static\_cast<float>(sum) / count;

} else {

return 0.0; // return 0 if no units are registered

}

}

std::string computeGrade(float avg) {

if (avg >= 70.0) {

return "A";

} else if (avg >= 60.0) {

return "B";

} else if (avg >= 50.0) {

return "C";

} else if (avg >= 40.0) {

return "D";

} else {

return "E";

}

}

void displayDetails() {

std::cout << "\nDetails for student:\n";

std::cout << "Reg no: " << regno << "\n";

std::cout << "Student name: " << name << "\n";

std::cout << "Student age: " << age << "\n";

std::cout << "Course: " << course << "\n";

std::cout << "Course id: " << course\_id << "\n";

}

void addMarks() {

std::cout << "Adding marks:\n";

getData();

}

};

int main() {

Student students[5];

for (int i = 0; i < 5; i++) {

std::cout << "Enter details for student " << i + 1 << ":\n";

students[i].getData();

}

// Display student details, add marks, and compute grades

for (int i = 0; i < 5; i++) {

students[i].displayDetails();

students[i].addMarks();

// Calculate average and compute grade

float avg = students[i].calculateAverage();

std::string grade = students[i].computeGrade(avg);

std::cout << "Average marks: " << avg << "\n";

std::cout << "Grade: " << grade << "\n";

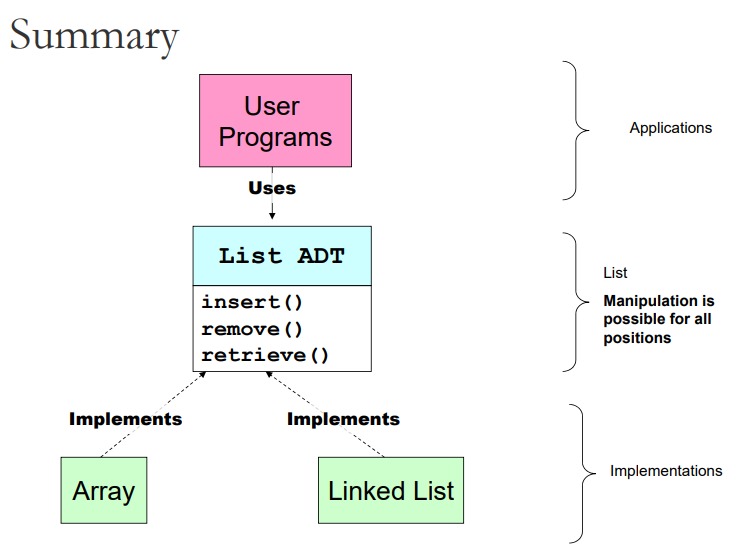
}

return 0;

}

**Task 4**

**Carry out a specification of ADT List using UML diagrams.**

****