PART A:

EDUCATION INSTITUTION PROPOSAL

Title: Smart Students System Using C++

Introduction:

Managing student performance efficiently is a key aspect of academic institutions. Traditional methods of student record-keeping and ranking require extensive manual work and are likely to make errors. The Smart Students System is designed to provide an automated solution for registering, sorting, searching, and displaying student records based on their GPA. This system will assist educational institutions in maintaining accurate student data, making informed academic decisions, and recognizing top-performing students without relying on manual documentation.

Problem Statement:

Currently, many academic institutions like The State University Of Zanzibar, Karume Institute of Science and Technology Zanzibar facing various challenges in managing student records effectively and in a good way. These challenges include:

- Manual Record-Keeping – inclined to mistake, loss or damage.
- Time-Consuming Sorting – ranking students manually is inefficient and is bad way.
- Difficulty in Searching – slow retrieval of student records.
- Human Errors – inaccurate data entry affects rankings.

Aim of the project:

The aim of this project is that, system will be developed using C++ and will run on Ubuntu nano. It will include features for registration, display, sorting, and searching. However, it will not support online databases in this initial version.

Objective:

On this part will deal with some of goal that are required to do.

General Objective:

- To develop a digital system that efficiently manages student records and ranks them based on their GPA.

Specific Objectives:

- Design a system for student registration.
- Implement sorting and searching functionalities.
- Store student data securely for future access.

Methodology:

The system will follow the Waterfall Model, including algorithm and pseudo code instruction for requirement analysis, design, implementation, and testing.

Waterfall Model means sequential software development methodology that structured of programming is directly linear where each instruction must be completed before moving to the next instruction.

Tools and Technologies

- Programming Language: C++
- IDE: Ubuntu nano
- Compiler: g++

Expected Result:

An expected result means output outcome that are intended to implement in the software program. On this expectation we look on before of the system established and after a system establish

I. Before the Smart Students System

Manual record-keeping makes retrieval slow.
Sorting and searching student data are inefficient.

II. After the Smart Students System

Automated data storage ensures quick retrieval.
Sorting and searching are fast and accurate.

Conclusion:

The Smart Students System will provide an efficient and automated solution for managing student records. By implementing this system, institutions can eliminate inefficiencies in student data management and make better academic decisions. The Smart Students is important and necessary because

- It saves time through automation.
- It improves accuracy in student ranking.
- It enables fast retrieval of student records.

PART B:

In this part, it includes Project Proposal designed Pseudo code and implementation of the Smart Students System.

<u>**Pseudo code**</u>

```
START
DECLARE smarts AS List of Smart Students
CALL loadSmarts(smarts)  // Load student records from file

REPEAT
    DISPLAY “********** Smart Student System **********”
    DISPLAY “1. Registration for Smart Student”
    DISPLAY “2. Display Smart Students”
    DISPLAY “3. Sort Smart Students by GPA”
    DISPLAY “4. Search Smart Student”
    DISPLAY “5. Exit”
    DISPLAY “Enter your choice: “

    INPUT choice

    SWITCH(choice)
      CASE 1:
        CALL addSmart(smarts)
      CASE 2:
        CALL displaySmarts(smarts)
      CASE 3:
        CALL sortSmarts(smarts)
      CASE 4:
        DISPLAY “Enter identification number: “
        INPUT id
        CALL searchSmart(smarts, id)
      CASE 5:
 EXIT PROGRAM
 DEFAULT:
DISPLAY “Invalid choice! Try again.”
END SWITCH
UNTIL FALSE
END
FUNCTION addSmart(smarts)
  DISPLAY “Enter Smart Student Name: “
  INPUT smart_name
  DISPLAY “Enter Identification Number: “
  INPUT id
  DISPLAY “Enter GPA: “
  INPUT gpa
  ADD (smart_name, id, gpa) TO smarts
  SAVE (smart_name, id, gpa) TO FILE
  DISPLAY “Smart Student added successfully!”
END FUNCTION

FUNCTION displaySmarts(smarts)
DISPLAY “ID | Smart Name | GPA”
FOR EACH student IN smarts
DISPLAY student.id, student.smart_name, student.gpa
END FOR
END FUNCTION

FUNCTION sortSmarts(smarts)
    SORT smarts BY GPA (Descending Order)
    DISPLAY “Smart Students sorted by GPA!”
END FUNCTION

FUNCTION searchSmart(smarts, id)
 FOR EACH student IN smarts
  IF student.id == id THEN
 DISPLAY “Smart Student Found: “, student.id, student.smart_name, student.gpa
RETURN
END
DISPLAY “Smart Student not found!”
FUNCTION loadSmarts(smarts)
OPEN FILE “smart_data.txt”
 IF FILE NOT FOUND
DISPLAY “No existing data found.”
RETURN
END
READ student records FROM FILE INTO smarts
CLOSE FILE
END
```

Implementation code

```
/*The Smart Students System is designed to provide an automated solution for
registering, sorting, searching, and displaying student smart records based on their
GPA. This system will assist educational institutions in maintaining accurate student
data, making informed academic decisions, and recognizing top-performing students
without relying on manual documentation.

*/

#include <iostream>//allow program to perform input output

#include <fstream>//allow file handling to operate, read or writing data to file

#include <vector>//allow dynamic array

#include <iomanip>//allow formatting output

#include <algorithm>//allow data structures on sorting and search



using std::cout;//it is used to display output

using std::cin;//it is used for taking user input

using std::endl;//it is used to insert new line

using std::string;//it used for handling smart name

using std::ofstream;//it is used to write file

using std::ifstream;//it is used to read from file

using std::vector;//it is used to store multiple smart objects for dynamically

using std::setw;//it is used formatting console output

using std::sort;//it is used to sort smart student based on GPA

using std::getline;//it ised to read name and spaces



//smart class definition

class Smart {

public:

    string smart_name;

    int id;
    float gpa;


    //constructor initializes smart with string supplied as argument
    Smart(string s, int i, float p) {
        smart_name = s;
        id = i;
        gpa = p;
    }


    //display a identification, smart name and gpa to the Smart user
    void display() {
        cout << setw(10) << id << setw(20) << smart_name << setw(10) << gpa << 
endl;
    }
};


//function that doing a registration of a new smart students
void addSmart(vector<Smart> &smarts) {
    string smart_name;
    int id;
    float gpa;


    cout << "Enter Smart Student Name: ";
    cin.ignore();
    getline(cin, smart_name);
    cout << "Enter identification number: ";
    cin >> id;
    cout << "Enter GPA: ";
    cin >> gpa;
```

```
}

    smarts.push_back(Smart(smart_name, id, gpa));


    //used to create file and to write information to files

    ofstream file("smart_data.txt", std::ios::app);

    file << id << "," << smart_name << "," << gpa << endl;

    file.close();


    cout << "Smart Student added successfully!\n";

}


//display identification number, Smart Name, and GPA to the  Smart user

void displaySmarts(vector<Smart> &smarts) {

    cout << setw(10) << "identification number" << setw(20) << "Smart Name" <<
setw(10) << "GPA" << endl;


    cout << "--------------------------------------------------\n";

    for (auto &s : smarts)

        s.display();

}


// used to sort elements in a range

void sortSmarts(vector<Smart> &smarts) {

    sort(smarts.begin(), smarts.end(), [](Smart a, Smart b) {

        return a.gpa > b.gpa; // Sorting in descending order

    });

    cout << "Smart Students sorted by GPA!\n";

}


//function used to search a smart stedent by the identintificatin number

void searchSmart(vector<Smart> &smarts, int id) {
    bool found = false;
    for (auto &s : smarts) {
        if (s.id == id) {
            cout << "Smart Student Found:\n";
            s.display();
            found = true;
            break;
        }
    }
    if (!found)
        cout << "Smart Student not found!\n";
}

//function to load smarts to file
void loadSmarts(vector<Smart> &smarts) {
    ifstream file("smarts_data.txt");
    if (!file) {
        cout << "No existing data found.\n";
        return;
    }

    string smart_name;
    int id;
    float gpa;
    while (file >> id) {
        file.ignore();
        getline(file, smart_name, ',');
        file >> gpa;
        smarts.push_back(Smart(smart_name, id, gpa));
    }
    file.close();
}


//function main begins program execution
int main() {
    vector<Smart> smarts;
    loadSmarts(smarts);


    int choice, id;
    while (true) {
        cout << "\n********** Smart Student System **********\n";


        cout << "1. Registration for Smart Student\n2. Display Smart Students\n3. Sort
Smart Students by GPA\n4. Search Smart Student\n5. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;


        switch (choice) {
            case 1: addSmart(smarts); break;
            case 2: displaySmarts(smarts); break;
            case 3: sortSmarts(smarts); break;
            case 4:
                cout << "Enter identification number: ";
                cin >> id;
                searchSmart(smarts, id);
                break;
            case 5: return 0;
            default: cout << "Invalid choice! Try again.\n";
        }
    }
return 0;//program ended successful
}
```

REFERENCES

- Deitel P. J & Deitel H. M (7th edition 2010): deitel c++ how to program.
- Stephen Prata, 6th.Edition (Oct.2011). C++Primer. Plus
- thinkCScpp