#include <stdio.h>

#include <string.h>

#define MAX\_STUDENTS 100

#define NAME\_LENGTH 50

#define REQUIRED\_ATTENDANCE 75

#define PASSING\_GRADE 7.0

// Códigos ANSI para cores

#define RED “\x1b[31m”

#define GREEN “\x1b[32m”

#define RESET “\x1b[0m”

// Estrutura para representar um aluno

Typedef struct {

Char name[NAME\_LENGTH];

Float grades[3];

Int attendance;

Float finalAverage;

Char status[20];

} Student;

// Função para calcular a média final (considerando as 2 maiores notas)

Float calculateFinalAverage(float grades[]) {

Float max1 = grades[0], max2 = grades[1];

// Identifica as duas maiores notas

If (grades[1] > grades[0]) {

Max1 = grades[1];

Max2 = grades[0];

}

If (grades[2] > max1) {

Max2 = max1;

Max1 = grades[2];

} else if (grades[2] > max2) {

Max2 = grades[2];

}

// Retorna a média das duas maiores notas

Return (max1 + max2) / 2.0;

}

// Função para determinar a situação do aluno (aprovado ou reprovado)

Void determineStatus(Student \*student) {

If (student->finalAverage >= PASSING\_GRADE && student->attendance >= REQUIRED\_ATTENDANCE) {

Strcpy(student->status, “Aprovado”);

} else {

Strcpy(student->status, “Reprovado”);

}

}

// Função para exibir o relatório dos alunos

Void printReport(Student students[], int count) {

Printf(“\nrelatorio final:\n”);

Printf(“%-30s %-10s %-15s\n”, “nome”, “media final”, “situacao”);

Printf(“------------------------------------------------------------\n”);

For (int i = 0; i < count; i++) {

If (strcmp(students[i].status, “Aprovado”) == 0) {

Printf(“%-30s %-10.2f %-15s\n”, students[i].name, students[i].finalAverage, GREEN “Aprovado” RESET);

} else {

Printf(“%-30s %-10.2f %-15s\n”, students[i].name, students[i].finalAverage, RED “Reprovado” RESET);

}

}

}

// Função principal

Int main() {

Student students[MAX\_STUDENTS];

Int studentCount;

// Pergunta a quantidade de alunos

Printf(“\n====================================\n”);

Printf(“Cadastro de Alunos – Sistema Escolar\n”);

Printf(“====================================\n”);

Printf(“Quantos alunos serao cadastrados? “);

Scanf(“%d”, &studentCount);

// Limpeza do buffer de entrada após o scanf

Getchar(); // Essa linha garante que a nova linha (\n) seja consumida após o scanf de um inteiro

// Verifica se o número de alunos está dentro do limite

If (studentCount <= 0 || studentCount > MAX\_STUDENTS) {

Printf(RED “Número inválido de alunos! Encerrando o programa.\n” RESET);

Return 1;

}

// Entrada de dados dos alunos

For (int i = 0; i < studentCount; i++) {

Printf(“\nAluno %d:\n”, i + 1);

// Lê o nome do aluno com fgets para permitir espaços

Printf(“Nome: “);

Fgets(students[i].name, NAME\_LENGTH, stdin);

Students[i].name[strcspn(students[i].name, “\n”)] = 0; // Remove a nova linha do final do nome

// Lê as notas do aluno

Printf(“Nota 1: “);

Scanf(“%f”, &students[i].grades[0]);

Printf(“Nota 2: “);

Scanf(“%f”, &students[i].grades[1]);

Printf(“Nota 3: “);

Scanf(“%f”, &students[i].grades[2]);

// Lê a frequência do aluno

Printf(“Frequência (em %%): “);

Scanf(“%d”, &students[i].attendance);

// Limpeza do buffer de entrada

Getchar(); // Consumir qualquer nova linha extra

// Calcula a média final e determina a situação

Students[i].finalAverage = calculateFinalAverage(students[i].grades);

determineStatus(&students[i]);

}

// Gera o relatório final

printReport(students, studentCount);

// Pausa para o usuário ver a saída

Printf(“\nPressione qualquer tecla para sair...”);

Getchar(); // Espera o usuário pressionar uma tecla antes de finalizar

Return 0;

}