structure to store food item-related information and provide solutions for the following tasks:

1. Find the difference in price between the most expensive and the cheapest food item.
2. Calculate the average calories of food items that cost more than $5.

#include <iostream>

#include <string>

#include <climits>

using namespace std;

const int MAX\_ITEMS = 6;

const int MAX\_INGREDIENTS = 5;

struct FoodItem {

int itemId;

string name;

double price;

int calories;

string ingredients[MAX\_INGREDIENTS];

int ingredientCount;

};

double findPriceDifference(FoodItem foodItems[], int size) {

double maxPrice = INT\_MIN;

double minPrice = INT\_MAX;

int i = 0;

while (i < size) {

if (foodItems[i].price > maxPrice) {

maxPrice = foodItems[i].price;

}

if (foodItems[i].price < minPrice) {

minPrice = foodItems[i].price;

}

I++;

}

return maxPrice - minPrice;

}

double findAverageCalories(FoodItem foodItems[], int size) {

double totalCalories = 0;

int count = 0;

int i = 0;

while (i < size) {

if (foodItems[i].price > 5) {

totalCalories += foodItems[i].calories;

count++;

}

i++;

}

if (count == 0) {

return 0;

} else {

return totalCalories / count;

}

}

int main() {

FoodItem foodItems[MAX\_ITEMS] = {

{1, "Burger", 8.99, 500, {"Bun", "Beef Patty", "Cheese", "Lettuce", "Tomato"}, 5},

{2, "Pizza", 12.50, 800, {"Dough", "Tomato Sauce", "Cheese", "Pepperoni", ""}, 4},

{3, "Salad", 4.99, 200, {"Lettuce", "Tomato", "Cucumber", "Olive Oil", ""}, 4},

{4, "Pasta", 7.75, 600, {"Pasta", "Tomato Sauce", "Parmesan Cheese", "", ""}, 3},

{5, "Sandwich", 6.50, 400, {"Bread", "Ham", "Cheese", "Lettuce", ""}, 4},

{6, "Sushi", 15.00, 300, {"Rice", "Nori", "Fish", "", ""}, 3}

};

double priceDifference = findPriceDifference(foodItems, MAX\_ITEMS);

double averageCalories = findAverageCalories(foodItems, MAX\_ITEMS);

cout << "Price difference between the most expensive and the cheapest food item: $" << priceDifference << endl;

cout << "Average calories of food items that cost more than $5: " << averageCalories << " calories" << endl;

return 0;

}

structure to store employee-related information and provide solutions for the following tasks:

1. Find the difference in performance scores between the highest and the lowest performing employees.
2. Calculate the average performance score of employees who have more than 5 years of experience.

#include <iostream>

#include <string>

#include <climits>

using namespace std;

const int MAX\_EMPLOYEES = 7;

struct Employee {

int employeeId;

string name;

int yearsOfExperience;

double performanceScore;

};

double findPerformanceScoreDifference(Employee employees[], int size) {

double maxScore = INT\_MIN;

double minScore = INT\_MAX;

int i = 0;

while (i < size) {

if (employees[i].performanceScore > maxScore) {

maxScore = employees[i].performanceScore;

}

if (employees[i].performanceScore < minScore) {

minScore = employees[i].performanceScore;

}

i++;

}

return maxScore - minScore;

}

double findAveragePerformanceScore(Employee employees[], int size) {

double totalScore = 0;

int count = 0;

int i = 0;

while (i < size) {

if (employees[i].yearsOfExperience > 5) {

totalScore += employees[i].performanceScore;

count++;

}

i++;

}

return (count == 0) ? 0 : totalScore / count;

}

int main() {

Employee employees[MAX\_EMPLOYEES] = {

{1, "Alice", 10, 85.5},

{2, "Bob", 3, 78.2},

{3, "Charlie", 6, 92.3},

{4, "David", 4, 88.1},

{5, "Eve", 8, 95.0},

{6, "Frank", 2, 73.4},

{7, "Grace", 7, 89.7}

};

double scoreDifference = findPerformanceScoreDifference(employees, MAX\_EMPLOYEES);

double averagePerformanceScore = findAveragePerformanceScore(employees, MAX\_EMPLOYEES);

cout << "Difference in performance scores between the highest and the lowest performing employees: " << scoreDifference << endl;

cout << "Average performance score of employees with more than 5 years of experience: " << averagePerformanceScore << endl;

return 0;

}

C++ structure to store product-related information and provide solutions for the following tasks:

1. Find the price difference between the most expensive and the cheapest product.
2. Calculate the average price of products that are in stock.

#include <iostream>

#include <string>

#include <climits>

using namespace std;

const int MAX\_PRODUCTS = 6;

struct Product {

int productId;

string name;

double price;

int stock;

};

double findPriceDifference(Product products[], int size) {

double maxPrice = INT\_MIN;

double minPrice = INT\_MAX;

int i = 0;

while (i < size) {

if (products[i].price > maxPrice) {

maxPrice = products[i].price;

}

if (products[i].price < minPrice) {

minPrice = products[i].price;

}

i++;

}

return maxPrice - minPrice;

}

double findAveragePriceInStock(Product products[], int size) {

double totalPrice = 0;

int count = 0;

int i = 0;

while (i < size) {

if (products[i].stock > 0) {

totalPrice += products[i].price;

count++;

}

i++;

}

return (count == 0) ? 0 : totalPrice / count;

}

int main() {

Product products[MAX\_PRODUCTS] = {

{101, "Laptop", 1200.50, 15},

{102, "Smartphone", 800.99, 50},

{103, "Tablet", 300.75, 30},

{104, "Smartwatch", 199.99, 75},

{105, "Headphones", 150.00, 0},

{106, "Keyboard", 50.50, 100}

};

double priceDifference = findPriceDifference(products, MAX\_PRODUCTS);

double averagePriceInStock = findAveragePriceInStock(products, MAX\_PRODUCTS);

cout << "Price difference between the most expensive and the cheapest product: $" << priceDifference << endl;

cout << "Average price of products that are in stock: $" << averagePriceInStock << endl;

return 0;

}