NAME:

KABEER AHMAD

ROLL NO:

SU92-BSCBM-F23-024

SECTION:

CB2A

SUBMITTED TO:

NOOR ULLAH KHAN

Question 1

1. Mark the following statements as true or false.

a. A double type is an example of a simple data type - True

b. A one-dimensional array is an example of a structured data type - True

c. The size of an array is determined at compile time. - True

d. Given the declaration:   
int list[10];   
the statement: list[5] = list[3] + list[2]; updates the content of the fifth component of the array list. (2) - True

e. If an array index goes out of bounds, the program always terminates in an error - False

f. The only aggregate operations allowable on int arrays are the increment and decrement operations. - False

g. Arrays can be passed as parameters to a function either by value or by reference. - True

h. A function can return a value of type array. - False

i. In C11, some aggregate operations are allowed for strings. (11, 12, 13) - True

j. The declaration: char name[16] = "John K. Miller"; declares name to be an array of 15 characters because the string "John K. Miller" has only 14 characters - True

k. The declaration: char str = "Sunny Day"; declares str to be a string of an unspecified length. - False

l. As parameters, two-dimensional arrays are passed either by value or by reference. (15, 16) - True

2. Consider the following declaration: (1, 2)

double currentBalance[91];

a. The array name - currentBalance

b. The array size - 91

c. The data type of each array component - double

d. The range of values for the index of the array - 0 to 90

e. The indices of the first, middle, and the last elements are 0, 45, and 90 respectively.

3. Identify error(s), if any, in the following array declarations. If a statement is incorrect, provide the correct statement. (1, 2)

a. int primeNum[99]; - Valid

b. int testScores[0]; - Invalid, array size cannot be 0. Correct statement: int testScores[1];

c. string names[60]; - Invalid in C++, should be std::string names[60];

d. int list100[0..99]; - Invalid syntax, use int list100[100];

e. double[50] gpa; - Invalid syntax, should be double gpa[50];

f. const double LENGTH = 26; double list[LENGTH - 1]; - Valid

g. const long SIZE = 100; int list[2 \* SIZE]; - Valid

Sure, here are the remaining questions with their answers:

4. Determine whether the following array declarations are valid. If a declaration is invalid, explain why. (1, 2)

a. int list[61]; - Valid

b. strings names[20]; - Invalid, strings is not a data type in C++. Should be std::string names[20];

c. double gpa[]; - Invalid, array size must be specified or initialized with values

d. double[-50] ratings[]; - Invalid, array size cannot be negative

e. string flowers[35]; - Invalid in C++, should be std::string flowers[35];

f. int SIZE = 10; double sales[2 \* SIZE]; - Valid

g. int MAX\_SIZE = 50; double sales[100 - 2 \* MAX\_SIZE]; - Valid

5. What would be a valid range for the index of an array of size 65? The valid range for the index is 0 to 64. The indices of the first, middle, and the last elements are 0, 32, and 64 respectively. (1, 3)

6. Write C11 statement(s) to do the following: (1, 2)

a. Declare an array alpha of 50 components of type int.

int alpha[50];

b. Initialize each component of alpha to -1.

for(int i=0; i<50; i++) alpha[i] = -1;

c. Output the value of the first component of the array alpha.

cout << alpha[0];

d. Set the value of the 25th component of the array alpha to 62.

alpha[24] = 62;

e. Set the value of the 10th component of alpha to three times the value of the 50th component of alpha plus 10.

alpha[9] = 3\*alpha[49] + 10;

f. Use a for loop to output the value of a component of alpha if its index is a multiple of 2 or 3.

for(int i=0; i<50; i++)

if(i%2==0 || i%3==0) cout << alpha[i] << " ";

g. Output the value of the last component of alpha.

cout << alpha[49];

h. Output the value of the alpha so that 15 components per line are printed.

for(int i=0; i<50; i++) {

cout << alpha[i] << " ";

if((i+1)%15 == 0) cout << endl;

}

i. Use a for loop to increment every other element (the even indexed elements).

for(int i=0; i<50; i+=2) alpha[i]++;

j. Create a new array, diffAlpha, whose elements are the differences between consecutive elements in alpha. What is the size of diffAlpha?

int diffAlpha[49];

for(int i=0; i<49; i++)

diffAlpha[i] = alpha[i+1] - alpha[i];

The size of diffAlpha is 49.

7. What is the output of the following program segment? (2)

0.00 0.50 4.00 27.00 128.00

378.00 127.00 -100.00 -227.00 -26.00

8. What is the output of the following C11 code? (2)

0 0 0 2 0 3 6 1

9. What is stored in list after the following C11 code executes?

list[] = {1, 2, 2, 4, 24, 120, 119, 5946}

10. What is stored in myList after the following C11 code executes?

myList[] = {2.50, 5.00, 15.00, 45.00, 22.50, 45.00}

11. Correct the following code so that it correctly sets the value of each element of myList to the index of the element.

int myList[10];

for (int i = 0; i < 10; i++)

myList[i] = i;

12. Correct the following code so that it correctly initializes and outputs the elements of the array intList.

int intList[5];

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

cin >> intList[i];

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

cout << intList[i] << " ";

cout << endl;

13. What is array index out-of-bound? An array index out-of-bound refers to when the index used to access an array element is outside the valid range of indices for that array. In C++, the language does not check for out-of-bound indices, which can lead to undefined behavior or errors during runtime.

14. Suppose that points is an array of 10 components of type double, and

points = {9.9, 9.6, 8.5, 8.5, 7.8, 7.7, 6.5, 5.8, 5.8, 4.6}

The output of the corrected code is:

points[0] and points[1] are out of order.

points[1] and points[2] are out of order.

points[3] and points[4] are out of order.

points[5] and points[6] are out of order.

points[7] and points[8] are out of order.

Corrected code:

for (int i = 0; i < 9; i++)

if (points[i] < points[i + 1])

cout << "points[" << i << "] and points[" << (i + 1) << "] are out of order." << endl;

15. Write C11 statements to define and initialize the following arrays.

a. double heights[10] = {5.2, 6.3, 5.8, 4.9, 5.2, 5.7, 6.7, 7.1, 5.10, 6.0};

b. int weights[7] = {120, 125, 137, 140, 150, 180, 210};

c. char specialSymbols[] = {'$', '#', '%', '@', '&', '!', '^'};

d. string seasons[4] = {"fall", "winter", "spring", "summer"};

16. Determine whether the following array declarations are valid. If a declaration is valid, determine the size of the array.

a. int list[] = {18, 13, 14, 16}; - Valid, size is 4

b. int x[10] = {1, 7, 5, 3, 2, 8}; - Valid, size is 10

c. double y[4] = {2.0, 5.0, 8.0, 11.0, 14.0}; - Invalid, too many initializers

d. double lengths[] = {8.2, 3.9, 6.4, 5.7, 7.3}; - Valid, size is 5

e. int list[7] = {12, 13, , 14, 16, , 8}; - Invalid, missing values must be at the end

f. string name[8] = {"John", "Lisa", "Chris", "Katie"}; - Invalid, string is not a data type in C++. Should be std::string.

17. Suppose that you have the following declaration:

int alpha[5] = {3, 12, -25, 72};

If this declaration is valid, what is stored in each of the five components of alpha?

alpha[0] = 3, alpha[1] = 12, alpha[2] = -25, alpha[3] = 72, alpha[4] = 0

18. Consider the following declaration.

int list[] = {3, 8, 10, 13, 6, 11};

a. Code to output the value of each component:

for(int i=0; i<6; i++) cout << list[i] << " ";

b. Code to set first 5 components as value - 3\*(value of previous component):

list[0] = 3;

for(int i=1; i<5; i++) list[i] = list[i] - 3\*list[i-1];

19. What is the output of the following C11 code?

5 -5 30 -165 630

20. What is the output of the following C11 code?

alpha: 1 3 5 7 9 8 11 14 17

beta: -2 -7 12 17 38 55 72 89 106 107 94 81 68 55

21. Consider the following overloaded function headings:

void printList(int list[], int size);

void printList(string sList[], int size);

Which of the following function calls is valid?

a. printList(ids, 50); - Valid

b. printList(birds, 70); - Valid

c. printList(unitPrice, 100); - Invalid, unitPrice is an array of double, not int

d. printList(ids, 75); - Invalid, size is 50

e. printList(birds, 50); - Invalid, size is 70

22. Suppose that you have the following function definition.

int find(int x, int y)

{

return (x + y - x \* y);

}

In the following statements, which function call is valid?

a. u = find(list1[0], v); - Valid

b. cout << find(list1[0], list2[9]) << endl; - Valid

c. cout << find(list1, list2) << endl; - Invalid, arguments should be int, not int arrays

d. for (int i = 0; i < 10; i++)

list3[i] = find(list1[i], list2[i]); - Valid

23. What is the output of the following C11 code?

1 35700.00 714.00

2 96800.00 1936.00

3 55000.00 1100.00

4 72500.00 1450.00

5 87700.00 1754.00

24. To store the number of cars sold by each salesperson in the array cars, output the total numbers of cars sold at the end of each month, and output the salesperson number selling the maximum number of cars:

int totalCars = 0, maxCars = 0, salespersonNum;

for(int i=0; i<10; i++) {

inFile >> cars[i];

totalCars += cars[i];

if(cars[i] > maxCars) {

maxCars = cars[i];

salespersonNum = i+1;

}

}

cout << "Total cars sold: " << totalCars << endl;

cout << "Salesperson " << salespersonNum << " sold the maximum of " << maxCars << " cars." << endl;

25. What is the output of the following program?

list: 90 60 20 0 10

26. What is the output of the following program?

Quantity Unit Cost Amount

3 15.00 45.00

5 20.00 100.00

2 5.00 10.00

8 3.00 24.00

1 75.00 75.00

Total due: $254.00

27. What is the output of the following C11 code?

1 3.50 10.70 235.22

2 7.20 6.50 292.66

3 10.50 12.00 785.40

4 9.80 10.50 644.14

5 6.50 8.00 326.80

28. When an array is passed as an actual parameter to a function, what is actually being passed? The address of the first element of the array is passed.

29. In C11, as an actual parameter, can an array be passed by value? No, arrays cannot be passed by value in C++. They are always passed by reference.

30. Sort the following list using the selection sort algorithm as discussed in this chapter. Show the list after each iteration of the outer for loop.

Initial list: 12, 50, 68, 30, 46, 5, 92, 10, 38

After iteration 1: 5, 50, 68, 30, 46, 12, 92, 10, 38

After iteration 2: 5, 10, 68, 30, 46, 12, 92, 50, 38

After iteration 3: 5, 10, 30, 68, 46, 12, 92, 50, 38

After iteration 4: 5, 10, 30, 38, 46, 12, 92, 50, 68

After iteration 5: 5, 10, 30, 38, 46, 12, 92, 50, 68

After iteration 6: 5, 10, 12, 38, 46, 30, 92, 50, 68

After iteration 7: 5, 10, 12, 38, 46, 30, 50, 92, 68

After iteration 8: 5, 10, 12, 38, 46, 30, 50, 68, 92

Sorted list: 5, 10, 12, 30, 38, 46, 50, 68, 92

31. What is the output of the following C11 program segment?

1 0 1 1 1 0 0 1 1

32. What is the output of the following C11 program segment?

Chris Johnson

Sheila Mann

Cindy Blair

33. Consider the following function heading

void modifyList(int list[], int length)

Yes, in the definition of the function modifyList, a range-based for loop can be used to process the elements of list. The range-based for loop iterates over the elements of the array, so it doesn't need the length of the array.

34. Given the declaration: char name[30];

mark the following statements as valid or invalid. If a statement is invalid, explain why

a. name = "Bill William"; - Invalid, array names cannot be assigned values

b. strcmp(name, "Tom Jackson"); - Valid

c. strcpy(name, "Jacksonville"); - Valid

d. cin >> name; - Invalid, cin >> reads until whitespace, cannot be used for strings

e. name[0] = 'K'; - Valid

f. bool flag = (name >= "Cynthia"); - Invalid, relational operators cannot be used with string literals and char arrays

35. Given the declaration:

char str1[20];

char str2[15] = "Fruit Juice";

mark the following statements as valid or invalid. If a statement is invalid, explain why

a. strcpy(str1, str2); - Valid

b. if (strcmp(str1, str2) == 0)

cout << " str1 is the same as str2" << endl; - Valid

c. if (strlen(str1) >= strlen(str2))

str1 = str2; - Invalid, array names cannot be assigned values

d. if (str1 > str2)

cout << "str1 > str2." << endl; - Invalid, relational operators cannot be used with string literals and char arrays

36. Given the declaration: char name[8] = "Shelly";

mark the following statements as "Yes" if they output Shelly. Otherwise, mark the statement as "No" and explain why it does not output Shelly.

a. cout << name; - Yes

b. for (int j = 0; j < 6; j++)

cout << name[j]; - No, it will output "Shell" as it stops at j=6 before the null terminator

c. int j = 0;

while (name[j] != '\0')

cout << name[j++]; - Yes

d. int j = 0;

while (j < 8)

cout << name[j++]; - No, it will output "Shelly\0\0\0" as it will print the null terminators after "Shelly"

37. Given the declaration:

char myStr[26];

char yourStr[26] = "Arrays and Strings";

a. strcpy(myStr, "Summer Vacation");

b. cout << strlen(yourStr);

c. strcpy(myStr, yourStr);

d. int compare

38. Assume the following declarations: (11, 12, 13)

char name[21];

char yourName[21];

char studentName[31];

a. cin >> name; - Invalid, cin reads until whitespace

b. cout << studentName; - Valid, will output the contents of studentName array

c. yourName[0] = '\0'; - Valid, sets the first character of yourName to null terminator

d. yourName = studentName; - Invalid, arrays cannot be assigned to each other

e. if (yourName == name) studentName = name; - Invalid, relational operators cannot be used to compare arrays

f. int x = strcmp(yourName, studentName); - Valid

g. strcpy(studentName, name); - Valid

h. for (int j = 0; j < 21; j++) cout << name[j]; - Valid, will output contents of name array

39. double matrix[4][3] = {{2.5, 3.2, 6.0}, {5.5, 7.5, 12.6}, {11.25, 16.85, 13.45}, {8.75, 35.65, 19.45}};

40. a. for(int i=0; i<3; i++) cin >> matrix[0][i];

b. for(int i=0; i<4; i++) cout << matrix[i][2] << " ";

c. for(int i=0; i<3; i++) cout << matrix[0][i] << " ";

cout << matrix[0][2];

d. matrix[3][2] += 13.6;

41. a. 30 components (5 rows \* 6 columns)

b. 5 rows

c. 6 columns

d. Summing across columns (nested loop on columns)

e. Summing across rows (nested loop on rows)

42. a. int alpha[10][20];

b. for(int i=0; i<10; i++)

for(int j=0; j<20; j++)

alpha[i][j] = 0;

c. for(int i=0; i<10; i++)

for(int j=0; j<20; j++)

if(i==0) alpha[i][j] = 1;

else alpha[i][j] = 2;

d. for(int i=0; i<10; i++) {

alpha[i][0] = 5;

for(int j=1; j<20; j++)

alpha[i][j] = 2\*alpha[i][j-1];

}

e. for(int i=0; i<10; i++) {

for(int j=0; j<20; j++)

cout << alpha[i][j] << " ";

cout << endl;

}

f. for(int j=0; j<20; j++) {

for(int i=0; i<10; i++)

cout << alpha[i][j] << " ";

cout << endl;

}

43. a. All elements of beta will be 0

b. beta[0][0]=0, beta[0][1]=1, beta[0][2]=2, beta[1][0]=1, beta[1][1]=2, beta[1][2]=3, beta[2][0]=2, beta[2][1]=3, beta[2][2]=4

c. beta[0][0]=0, beta[0][1]=0, beta[0][2]=0, beta[1][0]=0, beta[1][1]=1, beta[1][2]=2, beta[2][0]=0, beta[2][1]=2, beta[2][2]=4

d. beta[0][0]=0, beta[0][1]=1, beta[0][2]=2, beta[1][0]=1, beta[1][1]=2, beta[1][2]=3, beta[2][0]=2, beta[2][1]=3, beta[2][2]=0

e. beta[0][0]=0, beta[0][1]=0, beta[0][2]=0, beta[1][0]=1, beta[1][1]=2, beta[1][2]=0, beta[2][0]=2, beta[2][1]=1, beta[2][2]=2

44. a. void readIn(int arr[][10], int rows, int cols) {

for(int i=0; i<rows; i++)

for(int j=0; j<cols; j++)

cin >> arr[i][j];

}

readIn(flowers, 28, 10);

readIn(animals, 15, 10);

readIn(trees, 100, 10);

readIn(inventory, 30, 10);

b. int sumRow(int arr[][10], int rows, int cols) {

int sum = 0;

for(int j=0; j<cols; j++)

sum += arr[row][j];

return sum;

}

for(int i=0; i<28; i++)

cout << "Sum of row " << i << " of flowers: " << sumRow(flowers, i, 10) << endl;

c. void print(int arr[][10], int rows, int cols) {

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

for(int j=0; j<cols; j++)

cout << arr[i][j] << " ";

cout << endl;

}

}

print(flowers, 28, 10);

print(animals, 15, 10);

print(trees, 100, 10);

print(inventory, 30, 10);

Question 2: Programing Exercise

Question1:

#include <iostream>

using namespace std;

int main() {

    const int SIZE = 50;

    double alpha[SIZE];

    for (int i = 0; i < 25; i++) {

        alpha[i] = i \* i;

    }

    for (int i = 25; i < SIZE; i++) {

        alpha[i] = 3 \* i;

    }

    for (int i = 0; i < SIZE; i++) {

        cout << alpha[i] << " ";

        if ((i + 1) % 10 == 0) {

            cout << endl;

        }

    }

    return 0;

}

Question 2:

#include <iostream>

using namespace std;

int smallestIndex(int arr[], int size) {

    int minIndex = 0;

    int minVal = arr[0];

    for (int i = 1; i < size; i++) {

        if (arr[i] < minVal) {

            minVal = arr[i];

            minIndex = i;

        }

    }

    return minIndex;

}

int main() {

    int arr[] = {5, 2, 8, 1, 9, 3};

    int size = sizeof(arr) / sizeof(arr[0]);

    int minIndex = smallestIndex(arr, size);

    cout << "Index of smallest element: " << minIndex << endl;

    return 0;

}

Question 3:

#include <iostream>

using namespace std;

int lastLargestIndex(int arr[], int size) {

    int maxIndex = 0;

    int maxVal = arr[0];

    for (int i = 1; i < size; i++) {

        if (arr[i] >= maxVal) {

            maxVal = arr[i];

            maxIndex = i;

        }

    }

    return maxIndex;

}

int main() {

    int arr[] = {5, 8, 2, 8, 1, 8, 9};

    int size = sizeof(arr) / sizeof(arr[0]);

    int maxIndex = lastLargestIndex(arr, size);

    cout << "Index of last largest element: " << maxIndex << endl;

    return 0;

}

Question 4:

#include <iostream>

#include <fstream>

using namespace std;

int main() {

    ifstream infile("scores.txt");

    if (!infile) {

        cout << "Error opening file." << endl;

        return 1;

    }

    int scores[26];

    int count[8] = {0};

    int score;

    int i = 0;

    while (infile >> score) {

        scores[i++] = score;

    }

    int numScores = i;

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

        int range = scores[i] / 25;

        if (range < 0 || range > 7) {

            cout << "Score out of range: " << scores[i] << endl;

        } else {

            count[range]++;

        }

    }

    cout << "Score Range\tCount" << endl;

    cout << "0-24\t\t" << count[0] << endl;

    cout << "25-49\t\t" << count[1] << endl;

    cout << "50-74\t\t" << count[2] << endl;

    cout << "75-99\t\t" << count[3] << endl;

    cout << "100-124\t\t" << count[4] << endl;

    cout << "125-149\t\t" << count[5] << endl;

    cout << "150-174\t\t" << count[6] << endl;

    cout << "175-200\t\t" << count[7] << endl;

    infile.close();

    return 0;

}

Question 5:

#include <iostream>

#include <cctype>

using namespace std;

int main() {

    const int MAX\_LEN = 100;

    char str[MAX\_LEN];

    cout << "Enter a string: ";

    cin.getline(str, MAX\_LEN);

    for (int i = 0; str[i] != '\0'; i++) {

        str[i] = toupper(str[i]);

    }

    cout << "Uppercase string: " << str << endl;

    return 0;

}

Question 6:

#include <iostream>

#include <fstream>

#include <cstring>

using namespace std;

int main() {

    ifstream infile("test\_data.txt");

    if (!infile) {

        cout << "Error opening file." << endl;

        return 1;

    }

    const int NUM\_QUESTIONS = 20;

    char answers[NUM\_QUESTIONS + 1];

    infile.getline(answers, NUM\_QUESTIONS + 1);

    char studentId[10];

    char studentAnswers[NUM\_QUESTIONS + 1];

    int score;

    char grade;

    while (infile >> studentId >> studentAnswers) {

        score = 0;

        for (int i = 0; i < NUM\_QUESTIONS; i++) {

            if (studentAnswers[i] == answers[i]) {

                score += 2;

            } else if (studentAnswers[i] != ' ') {

                score -= 1;

            }

        }

        double percentage = static\_cast<double>(score) / (NUM\_QUESTIONS \* 2) \* 100;

        if (percentage >= 90) {

            grade = 'A';

        } else if (percentage >= 80) {

            grade = 'B';

        } else if (percentage >= 70) {

            grade = 'C';

        } else if (percentage >= 60) {

            grade = 'D';

        } else {

            grade = 'F';

        }

        cout << "Student ID: " << studentId << endl;

        cout << "Answers: " << studentAnswers << endl;

        cout << "Score: " << score << endl;

        cout << "Grade: " << grade << endl;

        cout << endl;

    }

    infile.close();

    return 0;

}

Question 7:

#include <iostream>

#include <cstring>

using namespace std;

int main() {

    const int NUM\_CANDIDATES = 5;

    char candidates[NUM\_CANDIDATES][20];

    int votes[NUM\_CANDIDATES];

    int totalVotes = 0;

    for (int i = 0; i < NUM\_CANDIDATES; i++) {

        cout << "Enter the last name of candidate " << i + 1 << ": ";

        cin >> candidates[i];

        cout << "Enter the number of votes for " << candidates[i] << ": ";

        cin >> votes[i];

        totalVotes += votes[i];

    }

    int winnerIndex = 0;

    for (int i = 1; i < NUM\_CANDIDATES; i++) {

        if (votes[i] > votes[winnerIndex]) {

            winnerIndex = i;

        }

    }

    cout << "Candidate\tVotes Received\t% of Total Votes" << endl;

    for (int i = 0; i < NUM\_CANDIDATES; i++) {

        double percentage = static\_cast<double>(votes[i]) / totalVotes \* 100;

        cout << candidates[i] << "\t\t" << votes[i] << "\t\t" << percentage << "%" << endl;

    }

    cout << "Total\t\t" << totalVotes << endl;

    cout << "The Winner of the Election is " << candidates[winnerIndex] << "." << endl;

    return 0;

}

Question 8:

#include <iostream>

using namespace std;

void inputArray(int alpha[]) {

    cout << "Enter 20 numbers:\n";

    for (int i = 0; i < 20; ++i) {

        cin >> alpha[i];

    }

}

void doubleArray(const int alpha[], int beta[]) {

    for (int i = 0; i < 20; ++i) {

        beta[i] = 2 \* alpha[i];

    }

}

void copyAlphaBeta(const int alpha[], const int beta[], int matrix[][4]) {

    for (int i = 0; i < 5; ++i) {

        for (int j = 0; j < 4; ++j) {

            matrix[i][j] = alpha[i \* 4 + j];

        }

    }

    for (int i = 5; i < 10; ++i) {

        for (int j = 0; j < 4; ++j) {

            matrix[i][j] = beta[(i - 5) \* 4 + j];

        }

    }

}

void printArray(const int arr[], int size) {

    for (int i = 0; i < size; ++i) {

        cout << arr[i] << " ";

        if ((i + 1) % 15 == 0)

            cout << endl;

    }

    cout << endl;

}

int main() {

    int alpha[20];

    int beta[20];

    int matrix[10][4];

    inputArray(alpha);

    doubleArray(alpha, beta);

    copyAlphaBeta(alpha, beta, matrix);

    cout << "Printing alpha:\n";

    printArray(alpha, 20);

    cout << "Printing beta:\n";

    printArray(beta, 20);

    cout << "Printing matrix:\n";

    for (int i = 0; i < 10; ++i) {

        printArray(matrix[i], 4);

    }

    return 0;

}

Question 9:

#include <iostream>

using namespace std;

int MONTHS = 12;

void getData(int temperatures[][2]) {

    cout << "Enter the highest and lowest temperatures for each month:\n";

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

        cout << "Month " << i + 1 << ":\n";

        cout << "High: ";

        cin >> temperatures[i][0];

        cout << "Low: ";

        cin >> temperatures[i][1];

    }

}

double averageHigh(int temperatures[][2]) {

    int total = 0;

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

        total += temperatures[i][0];

    }

    return static\_cast<double>(total) / MONTHS;

}

double averageLow(int temperatures[][2]) {

    int total = 0;

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

        total += temperatures[i][1];

    }

    return static\_cast<double>(total) / MONTHS;

}

int indexHighTemp(int temperatures[][2]) {

    int maxIndex = 0;

    for (int i = 1; i < MONTHS; ++i) {

        if (temperatures[i][0] > temperatures[maxIndex][0]) {

            maxIndex = i;

        }

    }

    return maxIndex;

}

int indexLowTemp(int temperatures[][2]) {

    int minIndex = 0;

    for (int i = 1; i < MONTHS; ++i) {

        if (temperatures[i][1] < temperatures[minIndex][1]) {

            minIndex = i;

        }

    }

    return minIndex;

}

int main() {

    int temperatures[MONTHS][2];

    getData(temperatures);

    double avgHigh = averageHigh(temperatures);

    double avgLow = averageLow(temperatures);

    int maxIndex = indexHighTemp(temperatures);

    int minIndex = indexLowTemp(temperatures);

    cout << "Average High: " << avgHigh << endl;

    cout << "Average Low: " << avgLow << endl;

    cout << "Highest Temperature: " << temperatures[maxIndex][0] << " (Month " << maxIndex + 1 << ")" << endl;

    cout << "Lowest Temperature: " << temperatures[minIndex][1] << " (Month " << minIndex + 1 << ")" << endl;

    return 0;

}

Question 10:

#include <iostream>

#include <cmath>

using namespace std;

int MAX\_NUMBERS = 100;

double calculateMean(int numbers[], int count) {

    int sum = 0;

    for (int i = 0; i < count; ++i) {

        sum += numbers[i];

    }

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

}

double calculateStandardDeviation(int numbers[], int count, double mean) {

    double variance = 0;

    for (int i = 0; i < count; ++i) {

        variance += pow(numbers[i] - mean, 2);

    }

    return sqrt(variance / count);

}

int main(){

    int numbers[MAX\_NUMBERS];

    int count;

    cout << "Enter the count of numbers (up to " << MAX\_NUMBERS << "): ";

    cin >> count;

    if (count > 0 && count <= MAX\_NUMBERS) {

        cout << "Enter " << count << " numbers:\n";

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

            cout << "Number " << i + 1 << ": ";

            cin >> numbers[i];

        }

        double mean = calculateMean(numbers, count);

        double standardDeviation = calculateStandardDeviation(numbers, count, mean);

        cout << "Mean (average): " << mean << endl;

        cout << "Standard Deviation: " << standardDeviation << endl;

    } else {

        cout << "Invalid count of numbers. Please enter a count between 1 and " << MAX\_NUMBERS << "." << endl;

    }

    return 0;

}

Question 11:

#include <iostream>

#include <string>

using namespace std;

int MAX\_DIGITS = 20;

void readNumber(string& numberString, int number[]) {

    cout << "Enter a positive integer of at most " << MAX\_DIGITS << " digits: ";

    cin >> numberString;

    int len = numberString.length();

    for (int i = 0; i < len; ++i) {

        number[len - i - 1] = numberString[i] - '0';

    }

}

void addNumbers(int num1[], int num2[], int sum[]) {

    int carry = 0;

    for (int i = 0; i < MAX\_DIGITS; ++i) {

        sum[i] = num1[i] + num2[i] + carry;

        carry = sum[i] / 10;

        sum[i] %= 10;

    }

}

void outputSum(int sum[]) {

    cout << "Sum of the numbers: ";

    bool leadingZero = true;

    for (int i = MAX\_DIGITS - 1; i >= 0; --i) {

        if (sum[i] != 0 || !leadingZero) {

            cout << sum[i];

            leadingZero = false;

        }

    }

    cout << endl;

}

int main() {

    string numberString1, numberString2;

    int num1[MAX\_DIGITS] = {0};

    int num2[MAX\_DIGITS] = {0};

    int sum[MAX\_DIGITS] = {0};

    readNumber(numberString1, num1);

    readNumber(numberString2, num2);

    addNumbers(num1, num2, sum);

    outputSum(sum);

    return 0;

}

Question 12:

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

const int NUM\_RUNNERS = 5;

const int NUM\_DAYS = 7;

void readData(string runners[], int miles[][NUM\_DAYS]) {

    ifstream inputFile("data.txt");

    if (!inputFile) {

        cerr << "Failed to open the file.\n";

        exit(1);

    }

    for (int i = 0; i < NUM\_RUNNERS; ++i) {

        inputFile >> runners[i];

        for (int j = 0; j < NUM\_DAYS; ++j) {

            inputFile >> miles[i][j];

        }

    }

    inputFile.close();

}

void calculateTotals(const string runners[], const int miles[][NUM\_DAYS], int totalMiles[], double averageMiles[]) {

    for (int i = 0; i < NUM\_RUNNERS; ++i) {

        totalMiles[i] = 0;

        for (int j = 0; j < NUM\_DAYS; ++j) {

            totalMiles[i] += miles[i][j];

        }

        averageMiles[i] = static\_cast<double>(totalMiles[i]) / NUM\_DAYS;

    }

}

void outputResults(const string runners[], const int miles[][NUM\_DAYS], const int totalMiles[], const double averageMiles[]) {

    cout << "Runner\t\tTotal Miles\tAverage Miles/Day\n";

    for (int i = 0; i < NUM\_RUNNERS; ++i) {

        cout << runners[i] << "\t\t" << totalMiles[i] << "\t\t" << averageMiles[i] << endl;

    }

}

int main() {

    string runners[NUM\_RUNNERS];

    int miles[NUM\_RUNNERS][NUM\_DAYS];

    int totalMiles[NUM\_RUNNERS];

    double averageMiles[NUM\_RUNNERS];

    readData(runners, miles);

    calculateTotals(runners, miles, totalMiles, averageMiles);

    outputResults(runners, miles, totalMiles, averageMiles);

    return 0;

}

Question 13:

#include <iostream>

#include <string>

using namespace std;

const int NUM\_STUDENTS = 10;

const int NUM\_TESTS = 5;

void readData(string names[], int scores[][NUM\_TESTS]) {

    cout << "Enter student names and their test scores:\n";

    for (int i = 0; i < NUM\_STUDENTS; ++i) {

        cin >> names[i];

        for (int j = 0; j < NUM\_TESTS; ++j) {

            cin >> scores[i][j];

        }

    }

}

void calculateGrades(const int scores[][NUM\_TESTS], char grades[], double averages[]) {

    for (int i = 0; i < NUM\_STUDENTS; ++i) {

        int total = 0;

        for (int j = 0; j < NUM\_TESTS; ++j) {

            total += scores[i][j];

        }

        averages[i] = static\_cast<double>(total) / NUM\_TESTS;

        if (averages[i] >= 90) {

            grades[i] = 'A';

        } else if (averages[i] >= 80) {

            grades[i] = 'B';

        } else if (averages[i] >= 70) {

            grades[i] = 'C';

        } else if (averages[i] >= 60) {

            grades[i] = 'D';

        } else {

            grades[i] = 'F';

        }

    }

}

void outputResults(const string names[], const char grades[], const double averages[]) {

    double classAverage = 0;

    cout << "Student\t\tAverage\tGrade\n";

    for (int i = 0; i < NUM\_STUDENTS; ++i) {

        cout << names[i] << "\t\t" << averages[i] << "\t" << grades[i] << endl;

        classAverage += averages[i];

    }

    classAverage /= NUM\_STUDENTS;

    cout << "Class Average: " << classAverage << endl;

}

int main() {

    string names[NUM\_STUDENTS];

    int scores[NUM\_STUDENTS][NUM\_TESTS];

    char grades[NUM\_STUDENTS];

    double averages[NUM\_STUDENTS];

    readData(names, scores);

    calculateGrades(scores, grades, averages);

    outputResults(names, grades, averages);

    return 0;

}

Question 14:

#include <iostream>

using namespace std;

const int ARRAY\_SIZE = 50;

void enterIntegers(int arr[]) {

    cout << "Enter 50 integers:\n";

    for (int i = 0; i < ARRAY\_SIZE; ++i) {

        cin >> arr[i];

    }

}

bool isSumOfTwo(int num, const int arr[]) {

    for (int i = 0; i < ARRAY\_SIZE; ++i) {

        for (int j = i + 1; j < ARRAY\_SIZE; ++j) {

            if (arr[i] + arr[j] == num) {

                return true;

            }

        }

    }

    return false;

}

void outputPairs(int num, const int arr[]) {

    cout << "Pairs where " << num << " is the sum of two other array elements:\n";

    for (int i = 0; i < ARRAY\_SIZE; ++i) {

        for (int j = i + 1; j < ARRAY\_SIZE; ++j) {

            if (arr[i] + arr[j] == num) {

                cout << arr[i] << " + " << arr[j] << endl;

            }

        }

    }

}

int main() {

    int arr[ARRAY\_SIZE];

    enterIntegers(arr);

    for (int i = 0; i < ARRAY\_SIZE; ++i) {

        if (isSumOfTwo(arr[i], arr)) {

            outputPairs(arr[i], arr);

        }

    }

    return 0;

}

Question 15:

#include <iostream>

using namespace std;

const int ARRAY\_SIZE = 50;

void enterIntegers(int arr[]) {

    cout << "Enter 50 integers:\n";

    for (int i = 0; i < ARRAY\_SIZE; ++i) {

        cin >> arr[i];

    }

}

void selectionSort(int arr[]) {

    for (int i = 0; i < ARRAY\_SIZE - 1; ++i) {

        int minIndex = i;

        for (int j = i + 1; j < ARRAY\_SIZE; ++j) {

            if (arr[j] < arr[minIndex]) {

                minIndex = j;

            }

        }

        if (minIndex != i) {

            swap(arr[i], arr[minIndex]);

        }

    }

}

bool isSumOfTwo(int num, const int arr[]) {

    int left = 0, right = ARRAY\_SIZE - 1;

    while (left < right) {

        int sum = arr[left] + arr[right];

        if (sum == num) {

            return true;

        } else if (sum < num) {

            left++;

        } else {

            right--;

        }

    }

    return false;

}

void outputPairs(int num, const int arr[]) {

    cout << "Pairs where " << num << " is the sum of two other array elements:\n";

    int left = 0, right = ARRAY\_SIZE - 1;

    while (left < right) {

        int sum = arr[left] + arr[right];

        if (sum == num) {

            cout << arr[left] << " + " << arr[right] << endl;

            left++;

            right--;

        } else if (sum < num) {

            left++;

        } else {

            right--;

        }

    }

}

int main() {

    int arr[ARRAY\_SIZE];

    enterIntegers(arr);

    selectionSort(arr);

    for (int i = 0; i < ARRAY\_SIZE; ++i) {

        if (isSumOfTwo(arr[i], arr)) {

            outputPairs(arr[i], arr);

        }

    }

    return 0;

}

Question 16:

#include <iostream>

#include<ctime>//for time

using namespace std;

int LOTTERY\_SIZE = 5;

void generateLotteryNumbers(int lottery[]) {

    srand(time(NULL));

    for (int i = 0; i < LOTTERY\_SIZE; ++i) {

        int num;

        do {

            num = rand() % 9 + 1;

        } while (lottery[i] == num);

        lottery[i] = num;

    }

}

bool containsDuplicates(int userNumbers[]) {

    for (int i = 0; i < LOTTERY\_SIZE - 1; ++i) {

        for (int j = i + 1; j < LOTTERY\_SIZE; ++j) {

            if (userNumbers[i] == userNumbers[j]) {

                return true;

            }

        }

    }

    return false;

}

void compareNumbers(int lottery[], int userNumbers[]) {

    int matchingDigits = 0;

    for (int i = 0; i < LOTTERY\_SIZE; ++i) {

        if (lottery[i] == userNumbers[i]) {

            ++matchingDigits;

        }

    }

    if (matchingDigits == LOTTERY\_SIZE) {

        cout << "Congratulations! You won the game!" << endl;

    } else {

        cout << "You didn't win. You matched " << matchingDigits << " digits." << endl;

    }

}

int main() {

    int lottery[LOTTERY\_SIZE];

    int userNumbers[LOTTERY\_SIZE];

    generateLotteryNumbers(lottery);

    cout << "Enter five distinct numbers between 1 and 9:" << endl;

    for (int i = 0; i < LOTTERY\_SIZE; ++i) {

        cin >> userNumbers[i];

    }

    if (containsDuplicates(userNumbers)) {

        cout << "Error: Input contains duplicate numbers." << endl;

        return 1;

    }

    compareNumbers(lottery, userNumbers);

    return 0;

}

Question 17:

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

const int NUM\_EMPLOYEES = 10;

const int DATA\_COLUMNS = 3;

void readData(string names[], int hours[][DATA\_COLUMNS]) {

    ifstream inputFile("employee\_data.txt");

    if (!inputFile) {

        cerr << "Failed to open the file.\n";

        exit(1);

    }

    for (int i = 0; i < NUM\_EMPLOYEES; ++i) {

        inputFile >> names[i];

        for (int j = 0; j < DATA\_COLUMNS; ++j) {

            inputFile >> hours[i][j];

        }

    }

    inputFile.close();

}

void calculateWeeklyPay(int hours[][DATA\_COLUMNS], double weeklyPay[]) {

    for (int i = 0; i < NUM\_EMPLOYEES; ++i) {

        int regularHours = hours[i][0] > 40 ? 40 : hours[i][0];

        int overtimeHours = hours[i][0] > 40 ? hours[i][0] - 40 : 0;

        double regularPay = regularHours \* hours[i][1];

        double overtimePay = overtimeHours \* hours[i][1] \* 1.5;

        weeklyPay[i] = regularPay + overtimePay;

    }

}

double computeAveragePay(const double weeklyPay[]) {

    double totalPay = 0;

    for (int i = 0; i < NUM\_EMPLOYEES; ++i) {

        totalPay += weeklyPay[i];

    }

    return totalPay / NUM\_EMPLOYEES;

}

void outputAboveAverage(const string names[], const double weeklyPay[], double averagePay) {

    cout << "Employees with pay greater than or equal to the average weekly pay:\n";

    for (int i = 0; i < NUM\_EMPLOYEES; ++i) {

        if (weeklyPay[i] >= averagePay) {

            cout << names[i] << endl;

        }

    }

}

void outputEmployeeData(const string names[], const int hours[][DATA\_COLUMNS], const double weeklyPay[]) {

    cout << "Employee data:\n";

    for (int i = 0; i < NUM\_EMPLOYEES; ++i) {

        cout << "Name: " << names[i] << ", Hours worked: " << hours[i][0] << ", Hourly pay rate: $" << hours[i][1]

                  << ", Weekly pay: $" << weeklyPay[i] << endl;

    }

}

int main() {

    string names[NUM\_EMPLOYEES];

    int hours[NUM\_EMPLOYEES][DATA\_COLUMNS];

    double weeklyPay[NUM\_EMPLOYEES];

    readData(names, hours);

    calculateWeeklyPay(hours, weeklyPay);

    double averagePay = computeAveragePay(weeklyPay);

    outputAboveAverage(names, weeklyPay, averagePay);

    outputEmployeeData(names, hours, weeklyPay);

    return 0;

}

Question 18:

#include <iostream>

#include <cstdlib> // for rand() and srand()

#include <ctime>   // for time()

using namespace std;

const int ROWS = 4;

const int COLS = 4;

const int NUM\_PAIRS = 8;

void initializeDeck(int deck[][COLS]) {

    int numPairs = 0;

    for (int num = 1; num <= NUM\_PAIRS; ++num) {

        for (int i = 0; i < 2; ++i) {

            int row, col;

            do {

                row = rand() % ROWS;

                col = rand() % COLS;

            } while (deck[row][col] != 0);

            deck[row][col] = num;

            numPairs++;

        }

    }

}

void printDeck(const int deck[][COLS], bool reveal) {

    cout << "  1 2 3 4" << endl;

    for (int i = 0; i < ROWS; ++i) {

        cout << i + 1 << " ";

        for (int j = 0; j < COLS; ++j) {

            if (reveal || deck[i][j] > 0) {

                cout << deck[i][j] << " ";

            } else {

                cout << "\* ";

            }

        }

        cout << endl;

    }

}

bool allCardsRevealed(const int deck[][COLS]) {

    for (int i = 0; i < ROWS; ++i) {

        for (int j = 0; j < COLS; ++j) {

            if (deck[i][j] == 0) {

                return false;

            }

        }

    }

    return true;

}

void playMemoryGame() {

    int deck[ROWS][COLS] = {0};

    initializeDeck(deck);

    int turns = 0;

    while (!allCardsRevealed(deck)) {

        printDeck(deck, false);

        int r1, c1, r2, c2;

        cout << "Enter the row and column numbers of two cards (separated by space): ";

        cin >> r1 >> c1 >> r2 >> c2;

        if (r1 < 1 || r1 > ROWS || c1 < 1 || c1 > COLS || r2 < 1 || r2 > ROWS || c2 < 1 || c2 > COLS) {

            cout << "Invalid input. Row and column numbers must be between 1 and " << ROWS << "." << endl;

            continue;

        }

        if (deck[r1 - 1][c1 - 1] == deck[r2 - 1][c2 - 1]) {

            cout << "Match!" << endl;

            deck[r1 - 1][c1 - 1] = 0;

            deck[r2 - 1][c2 - 1] = 0;

        } else {

            cout << "No match. Try again." << endl;

        }

        turns++;

    }

    cout << "Congratulations! You completed the game in " << turns << " turns." << endl;

    printDeck(deck, true);

}

int main() {

    srand(time(NULL));

    playMemoryGame();

    return 0;

}

Question 19:

#include <iostream>

#include <iomanip>

using namespace std;

const int ROWS = 13;

const int SEATS\_PER\_ROW = 6;

const char AVAILABLE = '\*';

const char OCCUPIED = 'X';

void initializeSeatingPlan(char seatingPlan[][SEATS\_PER\_ROW]) {

    for (int i = 0; i < ROWS; ++i) {

        for (int j = 0; j < SEATS\_PER\_ROW; ++j) {

            seatingPlan[i][j] = AVAILABLE;

        }

    }

}

void displaySeatingPlan(const char seatingPlan[][SEATS\_PER\_ROW]) {

    char seatLetter = 'A';

    cout << "  ";

    for (int i = 0; i < SEATS\_PER\_ROW; ++i) {

        cout << setw(2) << seatLetter++ << " ";

    }

    cout << endl;

    for (int i = 0; i < ROWS; ++i) {

        cout << "Row " << setw(2) << i + 1 << " ";

        for (int j = 0; j < SEATS\_PER\_ROW; ++j) {

            cout << setw(2) << seatingPlan[i][j] << " ";

        }

        cout << endl;

    }

}

void assignSeat(char seatingPlan[][SEATS\_PER\_ROW], int row, char seat) {

    int seatIndex = seat - 'A';

    if (seatingPlan[row - 1][seatIndex] == AVAILABLE) {

        seatingPlan[row - 1][seatIndex] = OCCUPIED;

        cout << "Seat assigned successfully." << endl;

    } else {

        cout << "Seat is already occupied. Please choose another seat." << endl;

    }

}

int main() {

    char seatingPlan[ROWS][SEATS\_PER\_ROW];

    initializeSeatingPlan(seatingPlan);

    int ticketType, row;

    char seat;

    while (true) {

        cout << "\n1. Assign Seat\n";

        cout << "2. Display Seating Plan\n";

        cout << "3. Exit\n";

        cout << "Enter your choice: ";

        cin >> ticketType;

        switch (ticketType) {

            case 1:

                cout << "Enter ticket type (1: First Class, 2: Business Class, 3: Economy Class): ";

                cin >> ticketType;

                if (ticketType < 1 || ticketType > 3) {

                    cout << "Invalid ticket type." << endl;

                    break;

                }

                cout << "Enter desired row number (1-13): ";

                cin >> row;

                if (row < 1 || row > ROWS) {

                    cout << "Invalid row number." << endl;

                    break;

                }

                cout << "Enter desired seat letter (A-F): ";

                cin >> seat;

                seat = toupper(seat);

                if (seat < 'A' || seat > 'F') {

                    cout << "Invalid seat letter." << endl;

                    break;

                }

                assignSeat(seatingPlan, row, seat);

                break;

            case 2:

                displaySeatingPlan(seatingPlan);

                break;

            case 3:

                cout << "Exiting program..." << endl;

                return 0;

            default:

                cout << "Invalid choice. Please try again." << endl;

        }

    }

    return 0;

}

Question 20:

#include <iostream>

using namespace std;

double computeAverageSpeed(double distance, double time) {

    if (time == 0) return 0;

    return distance / time;

}

int main() {

    const int MAX\_RECORDS = 100;

    double distances[MAX\_RECORDS];

    double times[MAX\_RECORDS];

    int numRecords = 0;

    cout << "Enter the number of records: ";

    cin >> numRecords;

    cout << "Enter distances and times:\n";

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

        cout << "Record " << i + 1 << " - Distance: ";

        cin >> distances[i];

        cout << "Record " << i + 1 << " - Time: ";

        cin >> times[i];

    }

    cout << "Average speed over successive time intervals:\n";

    for (int i = 0; i < numRecords - 1; ++i) {

        double intervalDistance = distances[i + 1] - distances[i];

        double intervalTime = times[i + 1] - times[i];

        double averageSpeed = computeAverageSpeed(intervalDistance, intervalTime);

        cout << "Interval " << i + 1 << ": " << averageSpeed << endl;

    }

    return 0;

}

Question 21:

#include <iostream>

using namespace std;

bool isPrime(int num) {

    if (num <= 1)

        return false;

    for (int i = 2; i \* i <= num; ++i) {

        if (num % i == 0)

            return false;

    }

    return true;

}

int findPrimeFactor(int num) {

    for (int i = 2; i \* i <= num; ++i) {

        if (num % i == 0 && isPrime(i))

            return i;

    }

    return -1;

}

int main() {

    const int MAX\_PRIMES = 1230;

    int primes[MAX\_PRIMES];

    int primeCount = 0;

    for (int num = 2; primeCount < MAX\_PRIMES; ++num) {

        if (isPrime(num)) {

            primes[primeCount++] = num;

        }

    }

    for (int num = 2; num <= 100'000'000; ++num) {

        if (isPrime(num)) {

            cout << num << " is prime.\n";

        } else {

            int factor = findPrimeFactor(num);

            cout << num << " is not prime. One of its prime factors is " << factor << ".\n";

        }

    }

    return 0;

}

Question 22:

#include <iostream>

using namespace std;

bool isPrime(int num) {

    if (num <= 1)

        return false;

    for (int i = 2; i \* i <= num; ++i) {

        if (num % i == 0)

            return false;

    }

    return true;

}

int findPrimeFactor(int num) {

    for (int i = 2; i \* i <= num; ++i) {

        if (num % i == 0 && isPrime(i))

            return i;

    }

    return -1;

}

int main() {

    for (int num = 2; num <= 100'000'000; ++num) {

        if (!isPrime(num)) {

            int factor = findPrimeFactor(num);

            cout << num << " = " << factor << " \* " << num / factor << endl;

        }

    }

    return 0;

}

Question 23:

#include <iostream>

#include <cstdlib> // for rand() and srand()

#include <ctime>   // for time()

using namespace std;

const int ROWS = 3;

const int COLS = 3;

const int HEIGHT = 3;

void fillArray(int array[ROWS][COLS][HEIGHT]) {

    for (int i = 0; i < ROWS; ++i) {

        for (int j = 0; j < COLS; ++j) {

            for (int k = 0; k < HEIGHT; ++k) {

                array[i][j][k] = rand() % 9 + 1;

            }

        }

    }

}

bool isMagicSquare(int array[ROWS][COLS][HEIGHT]) {

    int magicNumber = 0;

    for (int i = 0; i < ROWS; ++i) {

        for (int j = 0; j < COLS; ++j) {

            for (int k = 0; k < HEIGHT; ++k) {

                magicNumber += array[i][j][k];

            }

        }

    }

    magicNumber /= 3;

    for (int i = 0; i < ROWS; ++i) {

        for (int j = 0; j < COLS; ++j) {

            int rowSum = 0, colSum = 0, diagSum1 = 0, diagSum2 = 0;

            for (int k = 0; k < HEIGHT; ++k) {

                rowSum += array[i][j][k];

                colSum += array[j][i][k];

                if (i == j)

                    diagSum1 += array[i][j][k];

                if (i + j == 2)

                    diagSum2 += array[i][j][k];

            }

            if (rowSum != magicNumber || colSum != magicNumber)

                return false;

            if (i == 0 && j == 0 && (diagSum1 != magicNumber || diagSum2 != magicNumber))

                return false;

        }

    }

    return true;

}

int main() {

    srand(time(NULL));

    int array[ROWS][COLS][HEIGHT];

    int magicSquareCount = 0;

    int trials = 1000000;

    for (int t = 0; t < trials; ++t) {

        fillArray(array);

        if (isMagicSquare(array))

            magicSquareCount++;

    }

    cout << "Number of times the array is a magic square: " << magicSquareCount << endl;

    return 0;

}

Question 24:

#include <iostream>

#include <cstdlib> // for rand() and srand()

#include <ctime>   // for time()

using namespace std;

const int ROWS = 20;

const int HEIGHT = 3;

const int COLS = 20;

bool isPeak(int board[ROWS][HEIGHT][COLS], int row, int height, int col) {

    int neighbors[] = {-1, 0, 1};

    int countLess = 0, countGreater = 0;

    for (int i : neighbors) {

        for (int j : neighbors) {

            for (int k : neighbors) {

                int newRow = row + i;

                int newHeight = height + j;

                int newCol = col + k;

                if (newRow >= 0 && newRow < ROWS && newHeight >= 0 && newHeight < HEIGHT && newCol >= 0 && newCol < COLS) {

                    if (board[newRow][newHeight][newCol] < board[row][height][col]) {

                        countLess++;

                    } else if (board[newRow][newHeight][newCol] > board[row][height][col]) {

                        countGreater++;

                    }

                }

            }

        }

    }

    return (countLess == 26 || countGreater == 26);

}

int main() {

    srand(time(NULL));

    int board[ROWS][HEIGHT][COLS];

    for (int i = 0; i < ROWS; ++i) {

        for (int j = 0; j < HEIGHT; ++j) {

            for (int k = 0; k < COLS; ++k) {

                board[i][j][k] = rand() % 1000;

            }

        }

    }

    cout << "Peaks in the board:\n";

    for (int i = 0; i < ROWS; ++i) {

        for (int j = 0; j < HEIGHT; ++j) {

            for (int k = 0; k < COLS; ++k) {

                if (isPeak(board, i, j, k)) {

                    cout << "Peak at (" << i << ", " << j << ", " << k << ") - ";

                    if (board[i][j][k] > board[i-1][j][k] && board[i][j][k] > board[i+1][j][k] && board[i][j][k] > board[i][j-1][k] && board[i][j][k] > board[i][j+1][k] && board[i][j][k] > board[i][j][k-1] && board[i][j][k] > board[i][j][k+1]){

                      cout << "Maximum\n";

                    } else if (board[i][j][k] < board[i-1][j][k] && board[i][j][k] < board[i+1][j][k] && board[i][j][k] < board[i][j-1][k] && board[i][j][k] < board[i][j+1][k] && board[i][j][k] < board[i][j][k-1] && board[i][j][k] < board[i][j][k+1]){

                      cout << "Minimum\n";

                    }

                }

            }

        }

    }

    return 0;

}