C++

Chp 7

Exercises 7. Answer each of the following:

7.2 State whether the following are true or false. If the answer is false, explain why.

a) An array can store many different types of values. **False**.

Reason:

An array can store only values of the same type.

b) An array subscript should normally be of data type float. **False**.

Reason:

An array subscript should be an integer or an integer expression.

c) If there are fewer initializers in an initializer list than the number of elements in the ar-ray, the remaining elements are initialized to the last value in the initializer list. **False**.

Reason:

The remaining elements are initialized to zero.

d) It's an error if an initializer list has more initializers than there are elements in the array. **True**.

e) An individual array element that is passed to a function and modified in that function will contain the modified value when the called function completes execution. **False**.

Reason:

Individual elements of an array are passed by value. If the entire array is passed to a function, then any modifications to the elements will be reflected in the original.

7.3 Write one or more statements that perform the following tasks for and array called fractions:

a) Define a constant integer variable arraySize initialized to 10.

const int arraySize =10;

b) Declare an array with arraySize elements of type double, and initialize the elements to 0.

double fractions[arraySize] = {0.0} ;

c) Name the fourth element of the array. fractions[3]

d) Refer to array element 4.

fractions[4]

e) Assign the value 1.667 to array element 9. fractions[9]= 1.667;

f) Assign the value 3.333 to the seventh element of the array.

fractions[6]= 3.333;

g) Print array elements 6 and 9 with two digits of precision to the right of the decimal point, and show the output that is actually displayed on the screen.

cout << fixed << setprecision(2); cout << fractions [6] <<<< fractions[9] << endl;

Output: 3.33 1.67.

h) Print all the array elements using a for statement. Define the integer variable i as a control variable for the loop. Show the output.

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

cout << "fractions[" << i <<"]= “<< fractions[i]

<< endl;

Output: fractions[0] = 0.0 fractions[1] = 0.0 fractions [2] = 0.0 fractions [3] = 0.0 fractions [4] = 0.0 fractions [5] = 0.0 fractions [6] = 3.333 fractions [7] = 0.0 fractions [8] = 0.0 fractions[9] = 1.667 7.4

Answer the following questions regarding an array called table:

a) Declare the array to be an integer array and to have 3 rows and 3 columns. Assume that the constant variable arraySize has been defined to be 3.

int table [arraySize][arraySize];

b) How many elements does the array contain? Nine.

c) Use a for statement to initialize each element of the array to the sum of its subscripts. Assume that the integer variables i and j are declared as control variables.

for (i = 0 ; i < arraySize; i++) for(j = 0 j < arraySize; j++) table[i] [j]=i+j;

d) Write a program segment to print the values of each element of array table in tabular format with 3 rows and 3 columns. **Assume** that the array was initialized with the declaration int table[arraySize] [arraySize] = {{1,8}, {2, 4, 6), {5}}; and the integer variables i and j are declared as control variables. Show the output.

cout <<" [0] [1] [2]" << endl;

for (int i = 0; i< arraySize; i++) { cout <<'['<<i<<"] ": for (int j = 0 ;j < arraySize; j++) cout << setw(3) << table[i][j] <<" "; cout << endl; Output: [0] [1] [2] [0] 1 8 0 [1] 2 4 6 [2] 5 0 0

7.5 Find the error in each of the following program segments and correct the error:

a) #include ; **Error**: Semicolon at end of #include preprocessor directive.

Correction: Eliminate semicolon.

b) arraySize 10; // arraySize was declared const **Error**: Assigning a value to a constant variable using an assignment statement.

Correction: Initialize the constant variable in a const int arraySize declaration.

c) Assume that int b[10]={}; for (int i=0; i<= 10; i++) b[i] = 1;

Error: Referencing an array element outside the bounds of the array (b[10]).

Correction: Change the final value of the control variable to 9 or change <= to <.

d) Assume that int a[2][2]={{1, 2}, {3,4} }; a[1,1]=5; **Error**: Array subscripting done incorrectly. **Correction**: Change the statement to a[1][1] = 5;

Exercises 7.6 Fill in the blanks in each of the following:

a) The names of the four elements of array p **(int p[4];) are p[0], p[1], p[2], and p[3].**

b) Naming an array, stating its type and specifying the number of elements in the array is called **declaring** the array,

c) By convention, the first subscript in a two-dimensional array identifies an element's row ,and the second subscript identifies an **element's column**

d) An m-by-n array contains m rows, n columns and **m*n** elements.

e) The name of the element in row 3 and column 5 of array d is d[3][5].

7.7Determine whether each of the following is true or false. If false, explain why.

a) To refer to a particular location or element within an array, we specify the name of the array and the value of the particular element. **False**

Reason:

You must specify the subscript, not the value, to refer to an element.

b) An array definition reserves space for an array. **True**

Reason:

An array definition reserves space for the array

c) To indicate that 100 locations should be reserved for integer array p. you write the declaration p[100]; **False**

Reason:

You must include the data type. Correct form: int p[100];

e) A for statement must be used to initialize the elements of a 15-element array to zero. **False**

Reason

A for statement is commonly used but not required; initialization can be done in other ways.

f) Nested for statements must be used to total the elements of a two-dimensional array. **False**

Reason:

Nested for statements are typical, but not mandatory if handled differently.

7.8 Write C++ statements to accomplish each of the following:

a) Display the value of element 6 of character array f.

```
cout << f[6];
```

b) Input a value into element 4 of one-dimensional floating-point array b.

```
cin >> b[4];
```

c) Initialize each of the 5 elements of one-dimensional integer array g to 8.

```
for (int i = 0; i < 5; i++) g[i] = 8;
```

d) Copy array a into the first portion of array b. Assume double a[11], b[34]; for (int i = 0; i < 11; i++) b[i] = a[i];

e) Total and print the elements of floating-point array c of 100 elements.

```
float total = 0; for (int i = 0; i < 100; i++) total += c[i]; cout << total;
```

g) Determine and print the smallest and largest values contained in 99-element floating-point array w.

```
float min = w[0], max = w[0]; for (int i = 1; i < 99; i++) { if (w[i] < min) min = w[i]; if (w[i] > max) max = w[i]; } cout << min << " " << max;
```

7.9 Consider a 2-by-3 integer array t.

a) Write a declaration for t. int t[2][3];

b) How many rows does t have? 2

c) How many columns does t have? 3

d) How many elements does t have? 6

e) Write the names of all the elements in row 1 of t t[1][0], t[1][1], t[1][2]

f) Write the names of all the elements in column 2 of t. t[0][2], t[1][2]

g) Write a single statement that sets the element of t in the first row and second column to zero.

t[0][1] = 0; h)

Write a series of statements that initialize each element of t to zero. Do not use a loop.

t[0][0] = t[0][1] = t[0][2] = 0; t[1][0] = t[1][1] = t[1][2] = 0;

h) Write a nested for statement that initializes each element of t to zero.

for (int i = 0; i < 2; i++) for (int j = 0; j < 3; j++) t[i][j] = 0; j)

Write a statement that inputs the values for the elements of t from the keyboard.

for (int i = 0; i < 2; i++) for (int j = 0; j < 3; j++) cin >> t[i][j]; k)

Write a series of statements that determine and print the smallest value in array t.

int min = t[0][0]; for (int i = 0; i < 2; i++) for (int j = 0; j < 3; j++) if (t[i][j] < min) min = t[i][j]; cout << min; l)

Write a statement that displays the elements in row 0 of t.

for (int j = 0; j < 3; j++) cout << t[0][j] << " "; m)

Write a series of statements' that prints the array t in neat, tabular format. List the column subscripts as headings across the top and list the row subscripts at the left of each row.

cout << " 0 1 2\n"; for (int i = 0; i < 2; i++) { cout << i << " "; for (int j = 0; j < 3; j++) cout << t[i][j] << " "; cout << endl; } n)

Write a statement that totals the elements in column 3 of t. Total the elements in column 3 ✕ Invalid — column subscripts are 0, 1, 2. Column 3 does not exist.

7.13 Write single statements that perform the following one-dimensional array operations:

a) Initialize the 10 elements of integer array counts to zero. for (int i = 0; i < 10; i++) counts[i] = 0;

b) Add 1 to each of the 15 elements of integer array bonus. for (int i = 0; i < 15; i++) ++bonus[i];

c) Read 12 values for double array monthly Temperatures from the keyboard. for (int i = 0; i < 12; i++) cin >> monthlyTemperatures[i];

d) Print the 5 values of integer array bestScores in column format. for (int i = 0; i < 5; i++) cout << bestScores[i] << endl;

7.14 Find the error(s) in each of the following statement**s:**

a) Assume that: int a[3]; cout << a[1]<<" " <<a[2]<<" "<< a[3]<<endl; **Error**: a[3] is out of bounds Valid subscripts: a[0], a[1], a[2]

b) double f[3]= (1.1, 10.01, 100.001, 1000.0001); **Errors**: • Uses parentheses instead of braces • Too many initializers (4 values for 3 elements) **Correct** version: double f[3] = {1.1, 10.01, 100.001};

c) Assume that: double d[2][10]; d[1,9]=2.345; Incorrect subscript syntax

Correct statement: d[1][9] = 2.345; 7.16 **Label the elements** of a 3-by-5 two-dimensional array sales to indicate the order in which they're set to zero by the following program segment: for (row =0;row < 3; row++) for (column= 0; column<5; column++) sales[row] [column] =0; Order of assignment: sales[0][0] sales[0][1] sales[0][2] sales[0][3] sales[0][4] sales[1][0] sales[1][1] sales[1][2] sales[1][3] sales[1][4] sales[2][0] sales[2][1] sales[2][2] sales[2][3] sales[2][4]

7.18 What does the following program do?

#include using namespace std; int whatIsThis(int[], int); // function prototype int main() {const int arraySize =10; int a[arraySize]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; int result =whatIsThis(a ,arraySize); cout << "Result is" << result << endl; } // end main //what does this function do? int whatisThis (int b[], int size) if (size== 1) // base case return b[0]; else // recursive step return b[size- 1]+ whatIsThis(b, size- 1); }// end function whatisthis **Answer**: The program recursively calculates the sum of all elements in an integer array. **Explanation**: • The function whatIsThis: o Base case: when size == 1, returns b[0] o Recursive case: adds the last element to the sum of the remaining elements For the array: {1,2,3,4,5,6,7,8,9,10} **Result**: 1 + 2 + 3 + ... + 10 = 55 **Output**: Result is 55

✅ **Exercise 7.31** – Selection Sort (Recursive)**

Question **7.31**

A selection sort searches an array looking for the smallest element. Then, the smallest element is swapped with the first element of the array. The process is repeated for the subarray beginning with the second element of the array.

Write a recursive function `selectionSort` to perform this algorithm.

Program

```
#include <iostream>

using namespace std;

Void selectionSort(int arr[], int start, int size) {
  If (start == size – 1)
    Return;

  Int minIndex = start;
  For (int i = start + 1; i < size; i++) {
    If (arr[i] < arr[minIndex])
      minIndex = i;
  }
```

```
    Swap(arr[start], arr[minIndex]);
    selectionSort(arr, start + 1, size);
}

Int main() {
    Int arr[] = {64, 25, 12, 22, 11};
    Int size = 5;

    selectionSort(arr, 0, size);

    cout << “Sorted Array: “;
    for (int i = 0; i < size; i++)
        cout << arr[i] << “ “;
}
```

Output

```
Sorted Array: 11 12 22 25 64
```

✅ **Exercise 7.32 – Palindromes (Recursive)**

Question 7.32****

Write a recursive function `testPalindrome` that returns true if a string is a palindrome, and false otherwise.

Program

```cpp
#include <iostream>
#include <string>
Using namespace std;

Bool testPalindrome(string str, int start, int end) {
  If (start >= end)
    Return true;

  If (str[start] != str[end])
    Return false;

  Return testPalindrome(str, start + 1, end – 1);
}

Int main() {
```

```
  String word = “radar”;

  If (testPalindrome(word, 0, word.length() – 1))
    Cout << “Palindrome”;
  Else
    Cout << “Not Palindrome”;
}
```

Output

```
Palindrome
```

✅ **Exercise 7.33 – Linear Search (Recursive)**

Question **7.33**

Modify the program to use recursive function `linearSearch`. Return the index if found, otherwise return -1.

Program

```cpp
#include <iostream>
Using namespace std;

Int linearSearch(int arr[], int size, int key) {
   If (size == 0)
      Return -1;

   If (arr[size – 1] == key)
      Return size – 1;

   Return linearSearch(arr, size – 1, key);
}

Int main() {
   Int arr[] = {5, 3, 7, 1, 9};
   Int key = 7;

   Int result = linearSearch(arr, 5, key);

   Cout << “Index: “ << result;
}
```

Output

```
Index: 2
```

✅ **Exercise 7.34 – Eight Queens (Recursive)**

Question 7.34****

Modify the Eight Queens program to solve the problem recursively.

Program (Simplified)

```cpp
#include <iostream>
Using namespace std;

Int board[8];

Bool isSafe(int row, int col) {
   For (int i = 0; i < col; i++) {
      If (board[i] == row ||
```

```
        Abs(board[i] – row) == abs(i – col))
        Return false;
    }
    Return true;
}

Bool solve(int col) {
    If (col == 8)
        Return true;

    For (int i = 0; i < 8; i++) {
        If (isSafe(i, col)) {
            Board[col] = i;
            If (solve(col + 1))
                Return true;
        }
    }
    Return false;
}

Int main() {
    Solve(0);
    For (int i = 0; i < 8; i++)
        Cout << board[i] << “ “;
}
```

```

### **Output**

```
0 4 7 5 2 6 1 3
```

---

# ✅ **Exercise 7.35 – Print an Array (Recursive)**

### **Question 7.**35****

Write a recursive function `printArray` that prints an array.

---

### **Program**

```cpp
#include <iostream>
Using namespace std;

Void printArray(int arr[], int start, int end) {
  If (start > end)
    Return;
```
```

```
    Cout << arr[start] << " ";
    printArray(arr, start + 1, end);
}

Int main() {
    Int arr[] = {1, 2, 3, 4, 5};
    printArray(arr, 0, 4);
}
```

Output

```
1 2 3 4 5
```

✅ **Exercise 7.36 – Print String Backward**

Question 7.36****

Write a recursive function `stringReverse` that prints a string backward.

Program

```cpp
#include <iostream>
#include <string>
Using namespace std;

Void stringReverse(string str, int index) {
   If (index < 0)
      Return;

   Cout << str[index];
   stringReverse(str, index – 1);
}

Int main() {
   String word = “Hello”;
   stringReverse(word, word.length() – 1);
}
```

Output

```
olleH
```

```

---

# ✅ **Exercise 7.37 – Find Minimum in Array (Recursive)**

### **Question 7.37**

**Write a recur**sive function that returns the minimum element of an array.

```
#include <iostream>
Using namespace std;

Int recursiveMinimum(int arr[], int start, int end) {
  If (start == end)
    Return arr[start];

  Int min = recursiveMinimum(arr, start + 1, end);
  Return (arr[start] < min) ? arr[start] : min;
}

Int main() {
  Int arr[] = {4, 2, 9, 1, 5};
  Cout << “Minimum: “ << recursiveMinimum(arr, 0, 4);
}
```
```

Output

```
Minimum: 1
```

✅ **Exercise 7.**38** – Salesperson Salary Ranges (Vector)**

Program

```cpp
#include <iostream>
#include <vector>
Using namespace std;

Int main() {
   Vector<int> salary(9, 0);
   Int sales;

   While (true) {
      Cin >> sales;
      If (sales == -1)
         Break;
```

```
    Int index = (sales * 0.09 + 200) / 100 – 2;
    If (index >= 0 && index < 9)
      Salary[index]++;
  }

  For (int i = 0; i < 9; i++)
    Cout << “$” << (i+2)*100 << “: “ << salary[i] << endl;
}
```

✅ **Exercise 7.41 – Polling Program**

```
#include <iostream>
#include <string>
Using namespace std;

Int main() {
  String topics[5] = {
    “Environment”, “Education”, “Health”, “Poverty”, “Technology”
  };

  Int responses[5][10] = {0};

  Int rating;
```

```
    For (int i = 0; i < 5; i++) {
        Cout << “Rate “ << topics[i] << “ (1-10): “;
        Cin >> rating;
        Responses[i][rating – 1]++;
    }

    For (int i = 0; i < 5; i++) {
        Cout << topics[i] << “: “;
        For (int j = 0; j < 10; j++)
            Cout << responses[i][j] << “ “;
        Cout << endl;
    }
}
```
