

Instructions: This homework assignment focuses on the basics of C++ arrays. The answers to the following questions can be determined from Chapters 3 through 9 of the lecture notes and Chapters 11 and 12 of the text. Assume any `#include` directives, variable declarations, etc, which are needed to make the given code syntactically correct.

1) Given the declaration: `double Alpha[75];`

the logically valid range of index values for alpha is:

- | | | |
|-----------------|-----------------|------------------|
| 1) 0 through 75 | 3) 1 through 75 | 5) 1 through 76 |
| 2) 0 through 74 | 4) 1 through 74 | 6) none of these |
-

2) What is the output of the following program fragment?

```
int Gamma[3] = {5, 10, 15};  
  
for (int i = 0; i < 3; i++)  
    cout << Gamma[i] << ' ';
```

- | | |
|------------|---|
| 1) 5 10 15 | 4) 0 1 |
| 2) 5 10 | 5) Cannot be determined from the information given. |
| 3) 0 1 2 | 6) none of these |
-

3) Given the array declaration: `int status[10];`

which of the following loops correctly stores zeros at each valid index of the array `status[]`?

- 1)

```
for (int i = 0; i <= 10; i++)  
    status[i] = 0;
```
 - 2)

```
for (int i = 1; i < 10; i++)  
    status[i] = 0;
```
 - 3)

```
for (int i = 1; i <= 10; i++)  
    status[i] = 0;
```
 - 4)

```
for (int i = 0; i < 10; i++)  
    status[i] = 0;
```
 - 5)

```
for (int i = 1; i <= 11; i++)  
    status[i] = 0;
```
 - 6) none of these
-

4) You are writing a program to count the frequencies of characters that are read from a data file. (The computer uses the extended ASCII character set, which defines 256 different characters.) Which of the following array declarations is most appropriate, given that input characters will be used to index into the `freqCount[]` array?

- | | |
|--------------------------------------|--------------------------------------|
| 1) <code>char freqCount[256];</code> | 4) <code>int freqCount[char];</code> |
| 2) <code>char freqCount[int];</code> | 5) none of these |
| 3) <code>int freqCount[256];</code> | |
-

Questions 9 through 11 refer to the following incomplete program:

```
void FillEm(_____ arr1[], _____ arr2[], int length); // line 1
void Copy(_____ arr1[], _____ arr2[], int length);      // line 2

void main() {
    const int dim = 200;
    char alpha[dim];
    char beta[dim];

    FillEm(alpha, beta, dim); // Initialize arrays alpha and beta
    Copy(alpha, beta, dim);   // Copy all components of beta into alpha
}

void Copy(_____ arr1[], _____ arr2[], int length) {
    for (int Idx = 0; Idx < length; Idx++) {
        arr1[Idx] = arr2[Idx];
    }
    return;
}

void FillEm(_____ arr1[], _____ arr2[], int length) {
    // some initialization code goes here
}
```

9) What is the most appropriate way to fill the blank preceding the first formal parameter of `FillEm()` in line 1?

- | | |
|----------------------------|---------------------------|
| 1) <code>int&</code> | 5) <code>int</code> |
| 2) <code>char&</code> | 6) <code>const int</code> |
| 3) <code>char</code> | 7) all are equally good |
| 4) <code>const char</code> | 8) none are good |

10) What is the most appropriate way to fill the blank preceding the first formal parameter of `Copy()` in line 2?

- | | |
|----------------------------|---------------------------|
| 1) <code>int&</code> | 5) <code>int</code> |
| 2) <code>char&</code> | 6) <code>const int</code> |
| 3) <code>char</code> | 7) all are equally good |
| 4) <code>const char</code> | 8) none are good |

11) What is the most appropriate way to fill the blank preceding the second formal parameter of `Copy()` in line 2?

- | | |
|----------------------------|---------------------------|
| 1) <code>int&</code> | 5) <code>int</code> |
| 2) <code>char&</code> | 6) <code>const int</code> |
| 3) <code>char</code> | 7) all are equally good |
| 4) <code>const char</code> | 8) none are good |

12) Which of the following code fragments can be used to input values into a 3-element `int` array named `Alpha`?

- | | |
|--|--|
| 1) <code>cin >> Alpha[0] >> Alpha[1] >> Alpha[2];</code> | 4) <code>cin >> Alpha[0];
cin >> Alpha[1];
cin >> Alpha[2];</code> |
| 2) <code>cin >> Alpha;</code> | 5) All of them |
| 3) <code>for (i = 0; i < 3; i++)
cin >> Alpha[i];</code> | 6) 1 and 4 only |
| | 7) 1, 3 and 4 only |
| | 8) None of them |

13) You are writing a program to keep track of majors for a collection of students. Given the following declarations

```
const int MAXSTUDENTS = 100;
const int NUMMAJORS   = 4;
const string MAJORS[NUMMAJORS] = {"CS", "CpE", "Math", "Other"};
```

which of the following will properly declare two parallel arrays for storing student names and majors?

- | | |
|--|-----------------|
| 1) <code>string Name[MAXSTUDENTS];
string Major[NUMMAJORS];</code> | 4) All of them |
| 2) <code>string Name[MAXSTUDENTS];
string Major[MAXSTUDENTS];</code> | 5) 1 and 2 only |
| 3) <code>string Name[NUMMAJORS];
string Major[NUMMAJORS];</code> | 6) 1 and 3 only |
| | 7) 2 and 3 only |
| | 8) None of them |

For questions 14 through 17, assume that the following are global declarations in a program:

```
const int NOMATCH      = -1;
const int NUMMAJORS    = 4;
const string MAJORS[NUMMAJORS] = {"CS", "CpE", "Math", "Other"};
```

The following function should search the array `MAJORS[]` for the string `toFind`, and return the index of the matching string or an error indicator, as appropriate:

```
int FindMajor(_____ toFind) {           // line 1
    int Idx;                               // line 2
    for (Idx = 0; _____; Idx++) {     // line 3
        if (_____ == toFind)           // line 4
            return _____;           // line 5
    }
    return NOMATCH;                         // line 6
}
```

14) What is the most appropriate way to fill the blank preceding the formal parameter `toFind` in line 1?

- | | |
|-----------------------------------|----------------------------|
| 1) <code>string</code> | 4) All are equally good |
| 2) <code>string&</code> | 5) It should be left blank |
| 3) <code>const string&</code> | 6) None are good |

15) What is the most appropriate way to fill the blank in line 3?

- | | |
|--|----------------------------|
| 1) <code>Idx < NUMMAJORS</code> | 4) All are equally good |
| 2) <code>Idx <= NUMMAJORS</code> | 5) It should be left blank |
| 3) <code>Idx < MAJORS.length()</code> | 6) None are good |

16) What is the most appropriate way to fill the blank in line 4?

- | | |
|-----------------------------------|----------------------------|
| 1) <code>MAJORS[NUMMAJORS]</code> | 4) All are equally good |
| 2) <code>MAJORS[Idx]</code> | 5) It should be left blank |
| 3) <code>MAJORS</code> | 6) None are good |

17) What is the most appropriate way to fill the blank in line 5?

- | | |
|-----------------------------------|----------------------------|
| 1) <code>MAJORS[NUMMAJORS]</code> | 4) All are equally good |
| 2) <code>MAJORS[Idx]</code> | 5) It should be left blank |
| 3) <code>MAJORS</code> | 6) None are good |

18) Given the declarations below, how many components of type `double` does `Depth` have?

```
const int HEIGHT = 100;
const int WIDTH  = 50;
double Depth[HEIGHT][WIDTH];
```

- | | |
|------------|--|
| 1) 100 | 4) 100 * 50 |
| 2) 50 | 5) None, they are of type <code>int</code> . |
| 3) 99 * 49 | 6) None of these |

19) Given the declarations from question 18, the expression `Depth[5][3]` refers to:

- 1) the third element of the fifth row of `Depth`
- 2) the fifth element of the third row of `Depth`
- 3) the fourth element of the sixth row of `Depth`
- 4) the sixth element of the fourth row of `Depth`
- 5) The expression is not allowed.
- 6) None of these

20) Given the declarations below, the elements of `X` could be copied to the corresponding locations in `Y` by:

```
char X[100], Y[100];
```

- 1) the statement: `Y = X;`
- 2) the statement: `Y[] = X[];`
- 3) a user-defined function to which `X` and `Y` are passed
- 4) All of these.
- 5) 1 and 3 only
- 6) The elements of `Y` cannot possibly be copied into `X`.
- 7) None of these