Instructions: This homework assignment focuses on the basics of C++ arrays. The answers to the following questions can be determined from Chapters 3 through 8 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.

Opscan forms will be passed out in class. Write your name and code your ID number on the opscan form. Turn in your completed opscan at class on Monday Nov 5 or Tuesday Nov 6. Opscans will not be accepted at any other place or time.

1. Given the declaration: double Alpha[200];
   
   the logically valid range of index values for alpha is:

   1) 0 through 200
   2) 1 through 200
   3) 0 through 199
   4) 1 through 199
   5) 1 through 201
   6) None of these

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

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

   1) 5 10 15
   2) 15 10 5
   3) 10 15
   4) 15 10
   5) 5 10
   6) 10 5
   7) The code has a logic error.
   8) None of these

3. Given the declarations:
   int Status[10];
   int i;

   which of the following loops stores zeros at each valid index of the array Status[] without any logic errors?

   1) for (i = 1; i <= 10; i++)
      Status[i] = 0;
   6) 1 and 5 only
   2) for (i = 1; i <= 10; i++)
      Status[i - 1] = 0;
   7) 2 and 3 only
   3) for (i = 0; i <= 10; i++)
      Status[i] = 0;
   8) 2 and 4 only
   4) for (i = 0; i < 10; i++)
      Status[i] = 0;
   9) 3 and 4 only
   5) for (i = 1; i <= 11; i++)
      Status[i] = 0;
   10) 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, if the frequency counts are to be stored in the array freqCount[]?

   1) char freqCount[256];
   2) char freqCount[int];
   3) int freqCount[256];
   4) int freqCount[char];
   5) None of these
For questions 5 and 6, consider execution of the code fragment:

```c
int Arr[5];
int i;

for (i = 0; i < 5; i++) {
    Arr[i] = 2 * i + 3;
    if (i >= 3)
        Arr[i-1] = Arr[i] + 3;
}
```

5. What value will be contained in `Arr[2]`?

1) 3  
2) 5  
3) 7  
4) 9  
5) 10  
6) 11  
7) 12  
8) 14  
9) None of these

6. What value will be contained in `Arr[4]`?

1) 3  
2) 5  
3) 7  
4) 9  
5) 10  
6) 11  
7) 12  
8) 14  
9) None of these

7. What is the output of the following program fragment?

```c
int Alpha[5] = {100, 200, 300, 400, 500};
int i;

for (i = 3; i >= 0; i--)
    cout << Alpha[i] << ' ';
```

1) 400 300 200 100  
2) 500 400 300 200 100  
3) 500 400 300 200  
4) 4 3 2 1  
5) It cannot be answered from the information given.  
6) None of these

8. Given a 5000-element integer array `Beta[]`, which of the code fragments below could be used to print out the values of `Beta[0]`, `Beta[2]`, `Beta[4]`, and so forth up to `Beta[4998]`?

1) `for (int i = 0; i < 2500; i++)
   cout << Beta[2*i] << endl;`
2) `for (int i = 0; i < 5000; i++)
   cout << Beta[2*i] << endl;`
3) `for (int i = 0; i < 2500; i = i + 2)
   cout << Beta[i] << endl;`
4) `for (int i = 0; i < 5000; i = i + 2)
   cout << Beta[i] << endl;`

5) 1 and 3 only  
6) 1 and 4 only  
7) 2 and 3 only  
8) 2 and 4 only  
9) None of these
Questions 9 through 11 refer to the following incomplete program:

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

int 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
    return 0;
}

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) int&  
2) char&  
3) char   
4) const char  
5) int  
6) const int  
7) all are equally good  
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) int&  
2) char&  
3) char   
4) const char  
5) int  
6) const int  
7) all are equally good  
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) int&  
2) char&  
3) char   
4) const char  
5) int  
6) const int  
7) all are equally good  
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) `cin >> Alpha[0] >> Alpha[1]`  
   `>> Alpha[2];`  
2) `cin >> Alpha;`  
3) `for (i = 0; i < 3; i++)`  
   `cin >> Alpha[i];`  
4) `cin >> Alpha[0];`  
   `cin >> Alpha[1];`  
   `cin >> Alpha[2];`  
5) All of them  
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:

```cpp
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) `string Name[MAXSTUDENTS];`  
   `string Major[NUMMAJORS];`  
2) `string Name[MAXSTUDENTS];`  
   `string Major[MAXSTUDENTS];`  
3) `string Name[NUMMAJORS];`  
   `string Major[NUMMAJORS];`  
4) All of them  
5) 1 and 2 only  
6) 1 and 3 only  
7) 2 and 3 only  
8) None of these

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

```cpp
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:

```cpp
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) `string`  
2) `string&`  
3) `const string&`  
4) All are equally good  
5) It should be left blank  
6) None are good
15. What is the most appropriate way to fill the blank in line 3?

1) \( \text{Idx} < \text{NUMMAJORS} \)
2) \( \text{Idx} \leq \text{NUMMAJORS} \)
3) \( \text{Idx} < \text{MAJORS.length()} \)
4) All are equally good
5) It should be left blank
6) None are good

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

1) \( \text{MAJORS[NUMMAJORS]} \)
2) \( \text{MAJORS[Idx]} \)
3) \( \text{MAJORS} \)
4) All are equally good
5) It should be left blank
6) None are good

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

1) \( \text{MAJORS[NUMMAJORS]} \)
2) \( \text{MAJORS[Idx]} \)
3) \( \text{MAJORS} \)
4) All are equally good
5) It should be left blank
6) None are good

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

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

1) 100
2) 50
3) 99 * 49
4) 100 * 50
5) None, they are of type int.
6) None of these

19. Given the declarations from question 18, the expression \( \text{Depth[5][3]} \) 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:

```c
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