

Instructions: This homework assignment covers some of the basic C++ background you should have in order to take this course. The answers may be determined from the CS 1044 and CS 1704 notes, which are online at the respective course websites.

Opscan forms will be passed out on the first day of class and collected at the first class meeting during the second week of classes. Opscans will not be accepted at any other time. Mutilated opscans may be discarded.

I. Pointers

For questions 1 through 6, assume the variable declarations and initial memory layout shown below:

```
int a = 17,
    b = 42;
int *p = &a,
    *q = &b;
```

Identifier	Address
a	006AFDF4
b	006AFDF0
p	006AFDEC
q	006AFDE8

For questions 1 through 9, suppose the given statement is executed immediately after the initializations above, and determine the value of the indicated expression. Note that the questions are independent.

Chose from the following answers:

- | | | |
|-------------|-------------|------------------|
| 1) 006AFDF4 | 4) 006AFDE8 | 7) Unknown |
| 2) 006AFDF0 | 5) 17 | 8) None of these |
| 3) 006AFDEC | 6) 42 | |

	Statement to be executed	Expression
1.	*p = 27	p
2.	*p = 27	a
3.	*p = 27	&p
4.	p++	p
5.	p++	*p
6.	b = int(&b)	&b
7.	b = int(&b)	b
8.	b = int(&b)	q

9. Assume the variable declarations:

```
int Foo = 0;
int *ptr = &Foo;
```

Which of the following statements will change the value of Foo to 1?

- | | |
|-----------------|-------------------|
| 1) Foo++; | 6) 1 and 2 only |
| 2) ptr++; | 7) 1 and 4 only |
| 3) (*Foo)++; | 8) 2 and 4 only |
| 4) (*ptr)++; | 9) 3 and 4 only |
| 5) All of these | 10) None of these |

14. Assuming only the initial declarations given above, and execution of the correctly completed code given in question 11, what logical error(s) would result if the following statement were executed: `delete [] p;`
- 1) A dangling pointer would result (a pointer whose value is the address of memory that the program no longer owns).
 - 2) A memory leak would result (the program would own memory that it could no longer access).
 - 3) Both a dangling pointer and a memory leak would result.
 - 4) Neither a dangling pointer nor a memory leak, but some other logical error would result.
 - 5) No logical error would result.

15. Consider implementing a function to dynamically allocate an array of integers and reset all its elements to zero:

```
void ZeroIt(_____ A, const int Size) {
    A = new int[Size];
    for (int Idx = 0; Idx < Size; Idx++) {
        A[Idx] = 0;
    }
}
```

Which of the following choices for the blank preceding the formal parameter A is best?

- | | | |
|---------------------------|----------------------------|----------------------------------|
| 1) <code>int*</code> | 3) <code>const int*</code> | 5) <code>const int* const</code> |
| 2) <code>int*&</code> | 4) <code>int* const</code> | 6) All of the above |

For questions 16 through 18, assume the declarations:

```
struct Node {
    float Volume;
    Node* Next;
};
Node* headPtr;
```

Also assume that `headPtr` points to the first Node in a properly constructed linked list of 100 Nodes.

16. Which statement renders the head node (ONLY the head node) inaccessible?
- 1) `headPtr = headPtr->Next->Next;`
 - 2) `*headPtr = headPtr->Next;`
 - 3) `headPtr->Next = headPtr->Next->Next;`
 - 4) `headPtr->>(*Next) = headPtr->Next->Next;`
 - 5) None of these
17. Which statement changes the data element of the second node in the list?
- 1) `headPtr->Next = 42;`
 - 2) `headPtr->Next->Volume = 42;`
 - 3) `*(headPtr->Next->Volume) = 42;`
 - 4) `*(headPtr->Next) = 42;`
 - 5) None of these
18. Which code fragment deletes the first node in the list without rendering the remainder of the list inaccessible?
- 1) `delete headPtr;`
 - 2) `delete headPtr;`
`headPtr = new Node;`
 - 3) `delete headPtr;`
`headPtr = headPtr->Next;`
 - 4) `headPtr = headPtr->Next;`
 - 5) None of these

II. Miscellaneous Procedural Programming Issues

19. If a program will not compile which of the following must be true?

- | | |
|-------------------------------------|---------------------|
| 1) The syntax is not correct. | 6) 1 and 2 only |
| 2) The style is not correct. | 7) 1, 2, and 3 only |
| 3) The syntax is correct | 8) 3 and 4 only |
| 4) The design logic is not correct. | 9) None of these |
| 5) All of the above. | |

20. If a program compiles but does not produce correct output, which of the following must be true?

- | | |
|-------------------------------------|---------------------|
| 1) The syntax is not correct. | 6) 1 and 2 only |
| 2) The style is not correct. | 7) 1, 2, and 3 only |
| 3) The syntax is correct | 8) 3 and 4 only |
| 4) The design logic is not correct. | 9) None of these |
| 5) All of the above. | |

21. If a program compiles and produces correct output when given one sample input, which of the following must be true?

- | | |
|---------------------------------|---------------------|
| 1) The syntax is not correct. | 6) 1 and 2 only |
| 2) The style is not correct. | 7) 1, 2, and 3 only |
| 3) The syntax is correct | 8) 3 and 4 only |
| 4) The design logic is correct. | 9) None of these |
| 5) All of the above. | |

For the next two questions, consider the program:

```
#include <iostream>
#include <fstream>
using namespace std;

void main() {
    ifstream iFile;
    int int1;
    char char1;
    double double1;

    iFile.open("input.dat");
    iFile >> double1;
    while (iFile) {
        iFile.get(char1);
        iFile >> int1;
        cout << double1 << " " << char1 << " " << int1 << endl;

        iFile >> double1;
        iFile >> char1;
    }
}
```

Suppose the input file `input.dat` contains the following data, where `$` indicates the end of file character and each number is separated by a tab:

```
7      42.3  17      33      8.4    9.65$
```

22. What is the value of `char1` displayed on the second line of output?

- | | | |
|---------|--------|------------------|
| 1) '.' | 4) '2' | 7) None of these |
| 2) '\t' | 5) '3' | |
| 3) '1' | 6) '7' | |

23. What is the value of `int1` output on the third line of output?

- | | | |
|-------|-------|------------------|
| 1) 7 | 4) 17 | 7) 5 |
| 2) 42 | 5) 33 | 8) None of these |
| 3) 3 | 6) 65 | |

For the next three questions, assume the following rather uninteresting program:

```
#include <fstream>
using namespace std;

struct Point {
    double x, y;
};
const int MAXPOINTS = 100;

int ReadCorners(Point P[]);

void main() {
    Point Polygon[MAXPOINTS];
    int NumCorners;
    NumCorners = ReadCorners(Polygon);
}
// Line 1
// Line 2
// Line 3
```

The function `ReadCorners()` will read in coordinates for up to `MAXPOINTS` polygon corners, store them in the array of `Point` variables, and return the number of corners.

24. We want to add a function, called by `main()`, to the program to initialize the array of `Point` variables to hold dummy values. If the following implementation of such a function were used, with an appropriate prototype and a call added at Line 1 of `main()`, then:

```
void Init(const Point P[]) {
    Point Dummy = {0.0, 0.0};
    for (int Idx = 0; Idx < MAXPOINTS; Idx++)
        P[Idx] = Dummy;
}
```

- 1) The resulting program would not compile.
- 2) The resulting program would compile but would not behave as specified.
- 3) The resulting program would compile and behave as specified.
- 4) None of these

25. We want to modify the program to apply a transformation to the coordinates of each of the `Point` variables in the array. At Line 2 of `main()`, we will add the following loop:

```
for (int Idx = 0; Idx < NumCorners; Idx++)
    flipHorizontal(Polygon[Idx]);
```

We will add the following function implementation, with an appropriate prototype, to the program:

```
void flipHorizontal(Point P) {
    P.x = -1.0*P.x;
}
```

If these changes were made, then:

- 1) The resulting program would not compile.
 - 2) The resulting program would compile but would not behave as specified.
 - 3) The resulting program would compile and behave as specified.
 - 4) None of these
26. Finally, we want to add a function that will sort the array of `Point` variables in some manner (exactly what the ordering is doesn't matter for this question). Assume that the sort function will have the prototype below, and that the body of the sort function is implemented correctly:

```
void sortCorners(Point P[], int Size);
```

Suppose we add the following call at Line 3 of `main()`: `sortCorners(Polygon, MAXPOINTS);`

Then:

- 1) The array of `Point` variables will always be sorted correctly.
- 2) The array of `Point` variables will definitely never be sorted correctly.
- 3) The array of `Point` variables will be sorted correctly occasionally, but not always.
- 4) None of these

III. Some Class Issues

For questions 27 through 30, consider the classes `Track` and `Album`:

```
class Track {
private:
    string Title;
    int Length;
public:
    Track(string T, int L);
    string getTitle() const;
    int getLength() const;
};
```

```
class Album {
private:
    string Title;
    string Artist;
    int numTracks;
    Track* PlayList;
};
```

```
public:
    Album(string T, string A,
           int numTracks);
    bool AddTrack(const Track& T);
    Track getTrack(int Position) const;
    int getNumTracks() const;
    ~Album();
};

Album::Album(string T, string A, int nT) {
    Title = T;
    Artist = A;
    numTracks = nT;
    PlayList = new Track[nT];
}

Album::~Album() {
    delete [] PlayList;
}
```

Consider the following function, which computes the sum of the lengths of all the tracks on an album:

```
int Length(Album CD) {  
    int totalLength = 0;  
    for (int Idx = 0; Idx < CD.getNumTracks(); Idx++) {  
        totalLength += CD.getTrack(Idx).getLength();  
    }  
    return totalLength;  
}
```

27. The call `Length(myAlbum)` will have an unfortunate side effect (even though the body of the function is correct). What is that effect?
- 1) `myAlbum.numTracks` is changed.
 - 2) The destructor for `myAlbum` is invoked.
 - 3) The array of tracks, `myAlbum.PlayList[]`, is deleted.
 - 4) All of the above.
 - 5) 2 and 3 only
 - 6) None of these
28. Assuming that `myAlbum` is declared in the calling function, will the call `Length(myAlbum)` also cause a runtime exception at the end of the calling function?
- 1) Yes, definitely.
 - 2) No, definitely not.
 - 3) Perhaps yes, perhaps no, depending on factors not specified in the question.
29. Given that the interface and implementation of `Length()` cannot be changed, and that the use of global variables is unacceptable, which of the following actions is/are necessary to eliminate the difficulties cited in questions 27 and 28?
- 1) A deep copy constructor should be implemented for the class `Album`.
 - 2) A deep assignment operator overload should be implemented for the class `Album`.
 - 3) The destructor for the class `Album` should be removed from the class.
 - 4) All of the above.
 - 5) 1 and 2 only
 - 6) None of these
30. Aside from the action(s) you chose in question 29, which of the other actions should also be done in order to produce a solid implementation?
- 1) A deep copy constructor should be implemented for the class `Album`.
 - 2) A deep assignment operator overload should be implemented for the class `Album`.
 - 3) The destructor for the class `Album` should be removed from the class.
 - 4) All of the above.
 - 5) 1 and 2 only
 - 6) None of these