<u>Memory Leakage:</u> The design of the code fragment below causes a memory leak. Without breaking the otherwise correctness of the above code, answer the following five questions to fix the leak.

```
/*0*/   double group[1000];
/*1*/   double *gain;
/*2*/   double totalWeight = 0.0;
/*3*/   for (int Try = 0; Try < 1000; Try++) {
/*4*/           cout << “Please enter a distance: “;
/*5*/           gain = new double;
/*6*/           cin >> * gain;
/*7*/           totalWeight += gain;group[Try]=*gain;
        }
/*8*/   gain =NULL;
```

1.) What statement should be added?
   a. gain = NULL;
   b. delete gain;
   c. delete [] gain;
   d. delete double;
   e. delete totalWeight;
   f. None of the above.

2.) Where should the statement be added?
   a. Immediately after line 8
   b. Immediately after line 7
   c. Immediately after line 5
   d. Immediately after line 3
   e. Immediately after line 1
   f. Immediately before line 8
   g. None of the above.

3.) The memory lead could also be fixed by moving one statement.
   a. Move line 8 to immediately follow line 3.
   b. Move line 5 to immediately follow line 1.
   c. Move line 1 to immediately follow line 3.
   d. Two of the above would work.
   e. None of the above would work.

4.) A double occupies 8 bytes of memory. How many bytes of memory are leaked?
   a. 8
   b. 808
   c. 8008
   d. 7992
   e. 8000

5.) Suppose that the address of *group[0]* is at 1000. Suppose a variable *groupPtr* was a pointer to the array *group*, what would the address of ++*groupPtr* be?
   a. 1001
   b. 1002
   c. 1004
   d. 1008
   e. 1016
   f. None of the above.

6.) Which of the following is/are (an) example(s) of a memory leak?
   a. Garbage
   b. Aliases
   c. Dangling pointers
   d. Both a and b
   e. Both a and c
   f. Both b and c
   g. All of the above

<u>Pointers:</u>

7.) Which of the following correctly indicates a const pointer to non-constant data?
   a. const int * const x;
   b. const double * x;
   c. char * const x;
   d. string * x;

8.) **OMITED:**Which of the following is the most correct, assuming b is an int with value 7?
   a. int* aPtr=&b;a=5;
   b. int* cPtr=b; a++;
   c. int dPtr=&b; b*=5;
   d. int* ePtr = (&b-1)+1;
   e. int * fPtr = &b; a++;
   f. int gPtr =b; (*b)++;

9.) On which line is there an error in the following code?
   a. int x = 5, y = 7;
   b. int * const ptr = &x;
   c. *ptr = 7;
   d. ptr = &y;

<u>Method and Functions</u>

10.) A reference variable is always implicitly passed by value.
   a. True
   b. False
   c. HUH?

11.) Which of the following invokes the copy constructor?

a. MyListPtr= new List();
b. List *MyListPtr = new List();
c. MyListP = MyOtherList;
d. List MyList = MyOtherList;

12.) Which of the above invoke the assignment operator? **C**

Array Pointers: Use the following code to answer questions 10-11.

```
/*1*/    int BUGS[10] = {1,2,3,4,5,6,7,8,9,10};
/*2*/    nt *bugPointer=&BUGS[5];
/*3*/    bugPointer++;
/*4*/    cout << bugPointer << endl;
/*5*/    cout << BUGS[6] << endl;
/*6*/    int *bugPointer2=BUGS;
/*7*/    cout << bugPointer2[9] << endl;
/*8*/    cout << BUGS[9] << endl;
```

13.) Assume statements 6-8 are not in the above code. The above statements on lines 4 and 5 will print out the same value.

a. True
b. False
c. False, because of a compile error on line 3.

14.) Assuming line 3 is not in the code above, lines 7 and 8 will print the same value.

a. True
b. False
c. False, because of a compile error on line 6.

Abstract Data Types: Answer the following 5 questions with one of the following:

a.) Stacks
b.) Queues
c.) Lists
d.) DOS'
e.) DEQes

15.) I operate like a single lane of traffic at a stop light. **b**

16.) I operate like a pipe (for wires or water). **e**

17.) I operate like a narrow driveway with no place for a car to turn around. **a**

18.) I am like a cook who always serves his best tomatoes, but never serves a rotten one. **d**

19.) I can be compare to a blind man at a fence. **c**

20.) An abstract data type (ADT) describes both the behavior and implementation of a data structure.

a. True
b. False

21.) A namespace is used for modularization, so it is an example of encapsulation.

a. True
b. False

22.) Assume we have a linked list containing 5 elements with a *first* pointer. Each node has a *prev* and *next* pointer. Which of the following fragments of code would delete the second node from the linked list?

a.
```
Node* tmp= start->next;
delete tmp;
start->next=start->next->next;
start->next->next->prev=start;
```

b.
```
Node* tmp= start->next;
start->next=start->next->next;
start->next->next->prev=start;
delete tmp;
```

c.
```
Node* tmp= start->next;
start->next=tmp->next;
tmp->next->prev=tmp->prev;
delete tmp;
```

d. All of the above
e. a and b only
f. a and c only
g. b and c only
h. None of the above

23.) Assume we have an array of size 10 that we are implementing as a circular array for a queue implementation. Assume we have 1007 successful insertions and 999 successful deletes in some order such that the array was never full nor empty except before the first insertion. What is the value of front in the array? **8**

24.) What is the value of rear in the array above? **6**

25.) Assume a circular array has the values of 18 and 9 for the rear and front positions respectively. This array is:

a. Full
b. Empty
c. Only has one element

Matching:

26.)	nothrow **f**	a.) ptr + 3
27.)	"Address of" operator **e**	b.) *
28.)	Dereference operator **b**	c.) ptr[3]
29.)	Pointer-offset notation **a**	d.) 0
30.)	Pointer subscript notation **c**	e.) &
31.)	NULL **d**	f.) new

Word Problem

32.) The following values are enqueued onto a queue in the order given, then dequeued after all have been enqueued, pushed onto a drop out stack of size 5 at each dequeue. Each is then popped from the drop out stack and placed in a doubly linked list in which the current pointer is set to the head of the list after an insert and the insert operation inserts after the current pointer. What are the values in the linked list, sequentially, from head to tail? a,b,c,d,e,f,g,h,I **i,e,f,g,h**

Extra Credit

For 4 points of extra credit on this test, write out the interface for a double ended queue. Be as complete as possible for full credit. You may use the back of this (or any) page if necessary.