READ THIS NOW!

- Print your name in the space provided below. Code Form **B** on your Opscan. Check your SSN and Form Encoding!
- Choose the <u>single best answer</u> for each question — some answers may be partially correct. If you mark more than one answer, it will be counted wrong.
- Unless a question involves determining whether given C++ code is syntactically correct, assume that it is valid. The given code has been compiled and tested, except where there are deliberate errors. Unless a question specifically deals with compiler `#include` directives, you should assume the necessary header files have been included.
- Be careful to distinguish integer values from floating point (real) values (containing a decimal point). In questions/answers which require a distinction between integer and real values, integers will be represented without a decimal point, whereas real values will have a decimal point, [1704 (integer), 1704.0 (real)].
- The answers you mark on the Opscan form will be considered your official answers.
- When you have completed the test, sign the pledge at the bottom of this page and turn in the test.
- This is a closed-book, closed-notes examination. No calculators or other electronic devices may be used during this examination. You may not discuss (in any form: written, verbal or electronic) the content of this examination with any student who has not taken it. You must return this test form when you complete the examination. Failure to adhere to any of these restrictions is an Honor Code violation.
- There are 25 questions, equally weighted. The maximum score on this test is 100 points.

> ### Do not start the test until instructed to do so!

Print Name (Last, First) _____

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

_____
signature

I. Object Manipulations

Assume the following class declaration and implementation:

```
class IlliadUF {
private:
    bool* unix;
public:
    IlliadUF();
    IlliadUF(bool LordCrud);
    IlliadUF(const IlliadUF& Mike);
    bool getTF() const;
    void setTF(bool truth);
    ~IlliadUF();
};

IlliadUF::IlliadUF () {
    unix = new bool(true);
}

IlliadUF::IlliadUF (bool LordCrud) {
    unix = new bool(LordCrud);
}
```

```
bool IlliadUF::getTF() const {
    return(*unix);
}

void IlliadUF::setTF(bool truth) {
    *unix = truth;
}

IlliadUF::IlliadUF
        (const IlliadUF& Mike) {
    unix = new bool(*Mike.unix);
}

IlliadUF::~IlliadUF () {
    delete unix;
}
```

Given the following code:

```
void DustPuppy(IlliadUF Chief);

void main() {
    IlliadUF Miranda(false), AJ = Miranda;
    IlliadUF Cobb, Hillary(true);

    DustPuppy(Miranda);
    Cobb = Hillary;
    Hillary.setTF(!Hillary.getTF());

    cout << boolalpha; //stream modifier to output true/false for bools
    cout << "Contents of Miranda is:" << Miranda.getTF() << endl;      //LINE 1
    cout << "Contents of AJ is:"      << AJ.getTF()      << endl;      //LINE 2
    cout << "Contents of Cobb is:"    << Cobb.getTF()    << endl;      //LINE 3
    cout << "Contents of Hillary is:" << Hillary.getTF() << endl;      //LINE 4
}

void DustPuppy(IlliadUF Chief) { Chief.setTF(!Chief.getTF()); }
```

For the next 4 questions, select your answers from the following:

  1) true   3) Execution Error
  2) false   4) None of these

1.   What bool value is output by the call `Miranda.getTF()` in LINE 1 above?

2.   What bool value is output by the call `AJ.getTF()`   in LINE 2 above?

3.   What bool value is output by the call `Cobb.getTF()`   in LINE 3 above?

4.   What bool value is output by the call `Hillary.getTF()` in LINE 4 above?

5.   In the above code, immediately before `main()` goes out of scope, what is the total number of `IlliadUF` objects that has been dynamically allocated, (include in the count any that have been allocated and destructed).

```
(1) 1          (3) 3          (5) 5          (7) 7          (9) 0
(2) 2          (4) 4          (6) 5          (8) 8          (10)  None of the above
```

II. Recursion

Assume that the LinkNode and LinkList classes discussed in class have been implemented correctly and are available for use. The LinkList and LinkNode interfaces are given below:

```
#include "LinkNode.h"    // for node declaration
#include "Item.h"
class LinkList {
private:
   LinkNode* Head;      // points to head node in list
   LinkNode* Tail;      // points to tail node in list
   LinkNode* Curr;      // points to "current" node
public:
   LinkList(); //constructor
   LinkList::LinkList(const LinkList& Source);
   LinkList& LinkList::operator=
           (const LinkList& otherList);
   ~LinkList();//destructor
   bool isEmpty() const;
   bool inList() const;
   bool PrefixNode(const Item& newData);
   bool Insert(const Item& newData);
   bool Advance();
   void gotoHead();
   void gotoTail();
   bool DeleteCurrentNode();
   bool DeleteValue(const Item& Target);
   Item getCurrentData() const;
   void setCurrentData(const Item& newData);
};
```

```
//LinkNode.h
#include "Item.h"
class LinkNode {
private:
    Item      Data; //data "capsule"
    LinkNode* Next; //pointer next
node
public:
    LinkNode();
    LinkNode(const Item& newData);
    void setData(const Item& newData);
    void setNext(LinkNode* const
newNext);
    Item getData() const;
    LinkNode* getNext() const;
};
```

Assume that the Item type has been typedef'd to be equivalent to an `int` and that each node of the list holds one digit of an integer number. If the list stores a date then we might wish to code a recursive function to determine if the date is a palindrome. A palindrome is something that is the same forwards as backwards. For example, the date 10 02 2001 would be a palindromic date, (Europeans list the day first, so for them 20 02 2002 would be a palindromic date).

Given the following, incomplete, palidrome list class member functions:

```
    bool LinkList::ListPalindrome() {
       //setup function for Palindrome which does the real work
       if  ( ! isEmpty() ) {
          gotoHead();
          bool palin = Palindrome();
          for (; (Tail->getNext() != NULL); Tail = Tail->getNext()) ;
          return ( palin );
       } else
          return (false);
```

```
   } // ListPalindrome

   bool LinkList::Palindrome() {
      LinkNode* tmp;
      if (_____) //6.
         return true;
      if  (_____) //7.
         return (false);
      Advance();
      for (tmp = Curr; tmp->getNext() != Tail; tmp = tmp->getNext()) ;//null for
         _____; //8.
      return( Palindrome() ) ;
   } // Palindrome
```

6.   Select from the missing statements below to correctly fill in the blank for line numbered //**6.** in the above code to correctly satisfy the first base case of the Palindrome function?

   1) Head == NULL   4) Curr == Head
   2) Head == Tail   5) Curr == Tail
   3) Curr == NULL   6) None of the above

7.   Select from the missing statements below to correctly fill in the blank for line numbered //**7.** in the above code to correctly satisfy the second base case of the Palindrome function?

   1) Head->getData() == Tail->getData()
   2) Head->getData() != Tail->getData()
   3) getCurrentData() == Tail->getData()
   4) getCurrentData() != Tail->getData()

8.   Select from the missing statements below to correctly fill in the blank for line numbered //**8.** in the above code to correctly setup the recursive call to the Palindrome function?

   1) Head = tmp   4) Head = Curr
   2) Curr = tmp   5) Tail == Curr
   3) Tail = tmp   6) None of the above

9.   Which of the recursive problem solution methods is the `Palindrome()` function an example?

   1) Tail (going up) recursion
   2) Head (going down) recursion
   3) Middle Decomposition
   4) Edges & Center Decomposition
   5) Backtracking
   6) None of the above

III. Class Pointers

For the following 4 questions, assume the following declarations:

```
class   Article; //forward declaration
class   Article {
  private:
     Article* bond;
     bool     fact;
  public: //member functions
     void statement();
     ~Article();
};
```

```
//inside the Article member function statement
    Article* that = new Article;
    that->fact = true;
    that->bond = that;
    Article piece = *that;
    piece.bond->fact = false;
    piece.bond->bond = &piece;
    that = &piece;
    that->bond->bond = that;
```

10.   From inside the same member function as the above code,
      what is the **type**, (not value), of the expression at the right:

      ┌─────────────────────┐
      │ that->bond->bond   │
      └─────────────────────┘

      1) NULL   3) Article*   5) bond*
      2) Article   4) bond   6) None of the above

11.   From inside the same member function as the above code, which of the following statements could be used to change
      the Article object containing a false fact to a true fact (after the last statement above)?

      1) piece.fact = true;   4) piece->fact = true;
      2) that->fact = true;   5) piece->bond->fact = true;
      3) that->bond->fact = true;   6) None of the above

12. From inside the same member function as the above code, immediately
      before the function terminates, how many Article objects can be accessed?

      1) 1   3) 3   5) 0
      2) 2   4) 4   6) None of the above

13.   Considering just the code above, after the member function, statement(), has completed execution, (i.e. went out
      of scope), how many Article objects would the destructor be excuted upon?

      1) 1   3) 3   5) 0
      2) 2   4) 4   6) None of the above

IV. Separate Compilation

For the next three questions, consider a C++ program composed of three `cpp` files and three corresponding header files, as shown below, (the name of the file is in the first comment line of the file). All function calls are shown, as are all type declarations, function prototypes and *some* `include` directives. In the source and header files, there should be only one physical occurrence of a function prototype, and one physical occurrence of a type declaration. Do <u>not</u> assume that any preprocessor directives are used but not shown.

```
// main.h
class mClass {
   //. . .
};
```

```
// main.cpp
#include "main.h"
_____   //Line 14
_____   //Line 15
//. . .
void main() {
   Class2 C;
   Exam2(C);
   //. . .
}//end main

//. . .
   // mClass member Functions
```

```
// Exam2.h
//. . .
void Exam2(Class2& C2obj);
```

```
// Exam2.cpp
#include "Exam2.h"
//. . .
void Exam2(Class2& C2obj) {
   //. . .
}
```

```
// Class2.h
_____     //Line 16
//. . .
class Class2 {
  private:
     mClass MObj;
   //. . .
};
```

```
// Class2.cpp
#include "Class2.h"
//. . .
   // Class2 member Functions
```

14. If the organization shown above is used, and no preprocessor directives are added, which of the following `include` directives should replace the underscores at Line **14** above so that `main.cpp` can be successfully compiled?

   1) Nothing.
   2) `#include "main.h"`
   3) `#include "Exam2.h "`
   4) `#include "Class2.h"`
   5) `#include "Exam2.cpp"`
   6) `#include "Class2.cpp"`
   7) None of these.

15. If the organization shown above is used, and no preprocessor directives are added, which of the following `include` directives should replace the underscores at Line **15** above so that `main.cpp` can be successfully compiled?

   1) Nothing.
   2) `#include "main.h"`
   3) `#include "Exam2.h "`
   4) `#include "Class2.h"`
   5) `#include "Exam2.cpp"`
   6) `#include "Class2.cpp"`
   7) None of these.

16. If the organization shown above is used, and no preprocessor directives are added, which of the following `include` directives should replace the underscores at Line **16** above so that `Class2.cpp` can be successfully compiled?

   1) Nothing.
   2) `#include "main.h"`
   3) `#include "Exam2.h "`
   4) `#include "Class2.h"`
   5) `#include "Exam2.cpp"`
   6) `#include "Class2.cpp"`

IV. Separate Compilation (*continued*)

Consider the function call tree:
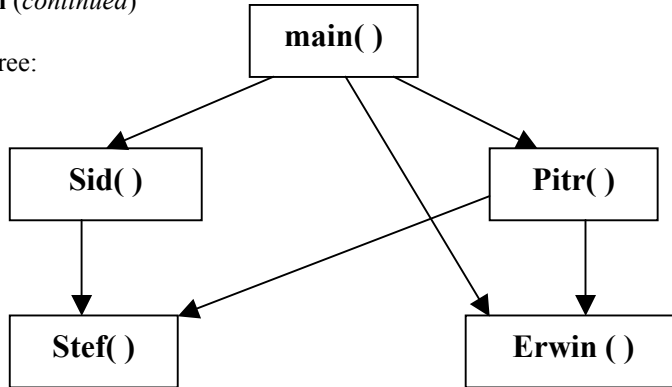
Assume that the software system is to be decomposed for compilation into three separate source files: `main.cpp`, `Sid.cpp`, and `Pitr.cpp`, and accompanying header files of the same names. The function definitions are to be placed in the various cpp files as shown below along with the corresponding code for the files.

FN definition locations

Definition for:	Goes in:
main( )	main.cpp
Sid( )	Sid.cpp
Stef( )	Sid.cpp
Pitr( )	Pitr.cpp
Erwin( )	Pitr.cpp

Scott separate compilation unit

```
//Pitr.h
void Pitr ( /* parameters */ );
```
```
// Pitr.cpp
#include "Pitr.h"
void Erwin( /* parameters */ );

void Pitr ( /* parameters */ ){
// Pitr's code
   Erwin();
   Stef();
}

void Erwin ( /* parameters */ ){
// Erwin's code
   Erwin( /* parameters */ );
}
```

Sid separate compilation unit

```
//Sid.h
void Sid ( /* parameters */ );
int greg;
```
```
// Sid.cpp
#include "Sid.h"

void Stef ( /* parameters */ );

void Sid ( /* parameters */ ){
// Sid's code
   Stef( /* parameters */ );
}

void Stef ( /* parameters */ ){
// Stef's code
}
```

main separate compilation unit

```
//main.h
/* main declarations */
```
```
//main.cpp
#include "main.h"
#include "Sid.h"

void main() {

   Sid  ( /* parameters */ );
   Pitr ( /* parameters */ );
   Erwin( /* parameters */ );

}
```

IV. Separate Compilation (*continued*)

Assume that there are no global type and no global constant declarations, (and also no global variables of course). Answer the following questions with respect to the above compilation organization and the goals of achieving information hiding and restricted scope:

17.   Assuming the partial code above was completed and contained no syntax errors, if **only** "`Pitr.cpp`" is **compiled** (not built) within Microsoft Visual C++, which of the following type of errors would occur:

```
1) Compilation error C2065: Erwin : undeclared identifier
2) Compilation error C2065: Stef : undeclared identifier
3) Compilation error C2001: missing main function.
4) No errors would be generated.
```

18.   Which of the following prototypes should be moved from its unit source.cpp file to the unit header.h file?

```
1) void Sid ( /* parameters */ );         3) void Erwin ( /* parameters */ );

2) void Pitr( /* parameters */ );          4) void main ( );
```

19.   In addition to the include directives listed above, where else **should** "`Pitr.h`" be included?

```
(1) main.h        (3) Sid.h        (5) Pitr.h
(2) main.cpp      (4) Sid.cpp      (6) nowhere else
```

20.   In addition to the include directives listed above, where else **should** "`Sid.h`" be included?

```
(1) main.h        (3) Pitr.cpp          (5) nowhere else
(2) Pitr.h        (4) Sid.h
```

21.   Assume all the code above was completed and contains no syntax or compilation errors. Further, assume that all of the header files have appropriate conditional compilation directives surrounding their contents. When the project containing the files is built within Microsoft Visual C++, which of the following linker errors would occur:

```
1) error LNK2015: multiple definitions for identifier 'Stef'
2) error LNK2015: multiple definitions for identifier 'Erwin'
3) Pitr.obj : error LNK2005: "int  greg" already defined in Sid.obj
4) No errors would be generated.
```

V. Linked List Class Manipulation

Consider the linked list class and list node declarations given below:

```
class ItemType {                        class LinkNode {
private:                                private:
    int Value;                             ItemType  Data;
public:                                    LinkNode* Next;
    ItemType();                         public:
    ItemType(int newValue);                LinkNode();
    void setValue(int newValue);           LinkNode(ItemType newData);
    int  getValue() const ;                bool setNext(LinkNode* newNext);
};                                         bool setData(ItemType  newData);
                                           ItemType  getData() const;
                                           LinkNode* getNext() const;
                                        };

                                        LinkNode *Head, *P, *Q;
```

Assume that the member functions above have been implemented correctly to carry out their intended task. Also, assume that operations have been executed to create the initial list structure below:
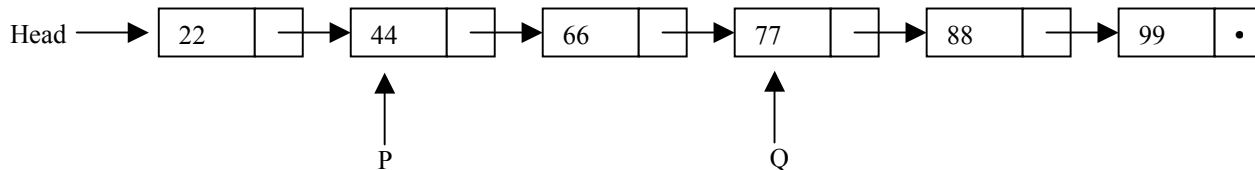
For the next 4 questions, select missing statements for the client, (not class), code segment below to transmogrify the above list into the list shown below:
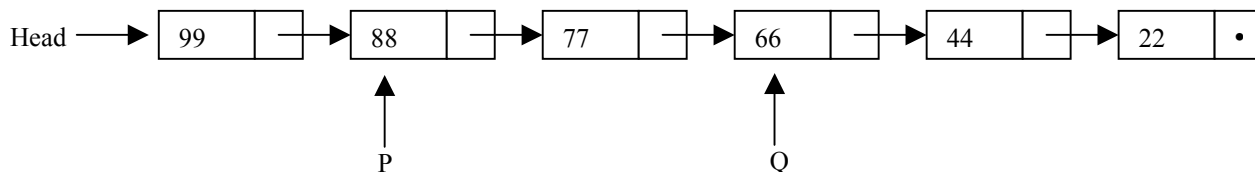

```
LinkNode* x = Head;                            //initialize x to first node
LinkNode* y = _____; //22.    //initialize y to last  node
LinkNode*  t;
if (x != NULL && y != NULL && x != y) {        //check trivial cases

  do {

    for (t=x; (t!=NULL && t->getNext()!= y); )  //hmmm what is this for doing?
        t = t->getNext();
    //swap *x & *y
    int s = _____; //23.      //assign  s to *x
    x-> _____; //24.          //assign *x to *y
    y->setData(s);
    x = x->getNext();                          //increment x
    y = _____; //25.               //decrement y

  } while ( x->getData().getValue() > y->getData().getValue() );

}//if
```

V. Linked List Class Manipulation (continued)

Select from the possible answers for the 4 questions given on the previous page.

1) `Q->Next->Next`	2) `Q->getNext()->getNext()`
3) `*x`	4) `x->getData()`
5) `x->getData().getValue()`	6) `setData(y->getData().getValue())`
7) `setData(y->Data.Value)`	8) `y->getNext()`
9) `t`	10) `t->getNext()`