Instructions: You will find it helpful to read the discussion of namespaces in the Deitel book. Experimentation may also be useful.

Opscan forms will be passed out in class on Tuesday, Oct 15. Write your name and code your ID number on the opscan form. Turn in your completed opscan at class on Tuesday Oct 22. No late opscans will be accepted.

For questions 1 and 2, assume the following header file and client code:

```
// A.h
#ifndef A_H
#define A_H

namespace A {
   const int MAX = 100;
};

#endif
```

```
// client code
#include <iostream>
using namespace std;
#include "A.h"
// added statements here:
. . .
int List[MAX];              // Line 1

int Sequence[A::MAX];       // Line 2
```

1. Which of the following statements, added immediately after the include section in the client code, will make the statement in Line 1 legal?

   1) `using namespace A;`
   2) `using A::MAX;`
   3) Either 1 or 2
   4) No additional statements are needed.
   5) None of these

2. Which of the following statements, added immediately after the include section in the client code, will make the statement in Line 2 legal?

   1) `using namespace A;`
   2) `using A::MAX;`
   3) Either 1 or 2
   4) No additional statements are needed.
   5) None of these

For questions 3 and 4, suppose that you want to use a commercial library of code that implements the following code, placing it in a header file named `Embroidery.h`.

```
namespace Embroidery {

   enum Color {RED, GREEN, BLUE};

   class string {
   private:
      Color Hue;
      int   Length;
   public:
      string();
      . . .
   };
};
```

Unfortunately, as you can see, the developer has made a rather unwise choice for the name of one class. In addition, you don't have access to the source code for the implementation of the class above, so you can't choose a more convenient name. Worse, you need to use the Standard `string` type in the same scope as the class declared above.

Suppose that you need to implement a C++ function that corresponds to the prototype below, where the first parameter is supposed to be a Standard `string` object and the second to be of the type declared in the `namespace Embroidery`. (Note that the code given here won't compile.)

```
void F(string Name, string Cord);
```

Assume that <string> and the header containing the `namespace` given above are both included, but that no `using` declarations of any kind have been given.

3. Which of the following will allow the code to compile, if placed appropriately in your code?

1) `using namespace std;`
2) `using namespace Embroidery;`
3) `using std::string;`
4) `using Embroidery::string;`
5) `std::string` preceding `Name`
6) `Embroidery::string` preceding `Cord`
7) 1 and 2 together
8) 3 and 4 together
9) 5 and 6 together
10) There is no way to do it.

4. Could the code above be made to compile (although not using the same technique, perhaps) if the class `string` had not been placed within a namespace by its developer?

1) Yes
2) No

For questions 5 and 6, consider the program:

```
#include <iostream>
#include <string>
using std::cin;
using std::cout;
using std::string;

const int A = 100;            // global

int main() {

   string A;                  // local
   cout << "Allocate memory? ";
   cin >> A;

   if ( A == "yes" ) {
      int *p = new int[____]; // allocation
   }

   return 0;
}
```

5. If the blank in the allocation statement is filled with `A`, which declaration of the identifier `A` will that bind to?

1) The global one
2) The local one
3) Neither

6. How should the blank in the allocation statement be filled in order to bind the dimension to the global declaration?

1) `std::A`
2) `::A`
3) `global::A`
4) `A`
5) It can't be done.
6) None of these

For questions 7 through 11, consider the class `Name` shown below, organized in a header file and implementation file. Each ellipsis indicates possible omitted code. Keep in mind that one goal is for `Name.cpp` to compile. You should keep in mind that `string` is a class, and you might want to experiment.

```
// Name.h
#ifndef NAME_H
#define NAME_H

#include <string>
. . .

class Name {

private:
   string First;
   string Middle;
   string Last;

public:
   Name();
   Name(const string& F,
        const string& M,
        const string& L);
   string fullName() const;
   string Initials() const;
};

#endif
```

```
// Name.cpp
. . .
using namespace std;

Name::Name() {
   First  = "";
   Middle = "";
   Last   = "";
}

Name::Name(const string& F,
           const string& M,
           const string& L) {
   First  = F;
   Middle = M;
   Last   = L;
}

string Name::fullName() const {

   string Full;
   if ( First != "" )
      Full = First;
   if ( Middle != "" )
      Full = Full + " " + Middle;
   if ( Last != "" )
      Full = Full + " " + Last;
   return Full;
}

string Name::Initials() const {

   string Init;
   if ( First != "" )
      Init += First[0];
   if ( Middle != "" )
      Init += Middle[0];
   if ( Last != "" )
      Init += Last[0];
   return Init;
}
```

7. Where could the following preprocessor directive be placed (usefully): `#include <string>`

   1) Preceding the class declaration in `Name.h`
   2) Following the class declaration in `Name.h`
   3) At the beginning of `Name.cpp`
   4) Either 1 or 2 or 3
   5) Either 1 or 2 but not 3
   6) It is not needed.
   7) None of these

8. Given that the include directive for `<string>` is placed at the beginning of `Name.h` as shown, what is needed after that directive in order to achieve compilation?

1) `using namespace std;`
2) `using std::string;`
3) `#include "Name.cpp"`
4) Either 1 or 2 or 3
5) Either 1 or 2
6) Either 2 or 3
7) Nothing else is needed.
8) None of these

9. Assuming that choice 2 from question 8 is used, is it necessary to include the directive "`using namespace std;`" within the file `Name.cpp` as shown above?

1) Yes, otherwise there will be compilation errors within the implementation of `fullName()`.
2) No, `Name.cpp` would compile without that directive.
3) None of these

10. Assuming the code organization given above, is it necessary to have an include directive for `Name.h` within `Name.cpp`?

1) Yes, in order to include `<string>` in `Name.cpp`.
2) Yes, in order to include the declaration of the class `Name` in `Name.cpp`.
3) Both 1 and 2
4) No, the include directive is not necessary.
5) None of these

11. The `#ifndef/#define/#endif` construct is used in `Name.h`. Is that necessary in order for `Name.cpp` to compile properly?

1) Yes
2) No

For questions 12 and 13, consider the header files for the classes `StudentID` and `RoomAssignment`:

```
// StudentID.h
#ifndef STUDENTID_H
#define STUDENTID_H

. . .

class StudentID {
private:
   Name   stuName;
   string SSN;
   string PID;

public:
   StudentID();
   StudentID(const Name& N,
             const string& S,
             const string& P);
};

#endif
```

```
// RoomAssignment.h
#ifndef ROOMASSIGNMENT_H
#define ROOMASSIGNMENT_H

. . .

class RoomAssignment {
private:
   Name   Resident1;
   Name   Resident2;
   string Dorm;
   int    Room;

public:
   RoomAssignment();
   void setFirst(const Name& N);
   void setSecond(const Name& N);
   void setDorm(const string& D);
   void setRoom(int R);
};

#endif
```

(The implementations exist, but the details are irrelevant for the following questions.)

12. What include directives are needed in `StudentID.h`?

1) `#include <string>`
2) `#include "Name.h"`
3) Both 1 and 2
4) None are needed.
5) None of these

13. Consider implementing a program that uses both classes; suppose that the beginning of that code is:

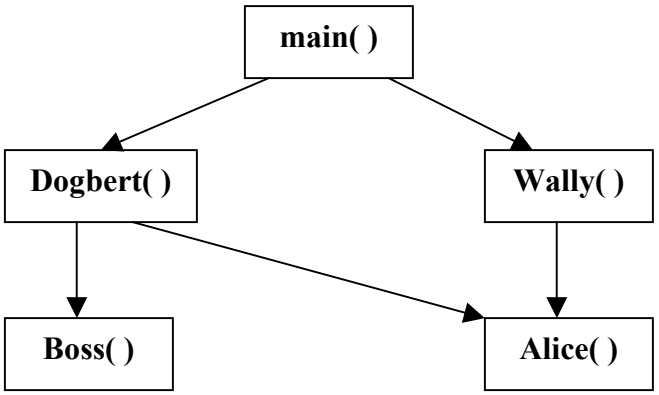
```
. . .
#include "StudentID.h"
#include "RoomAssignment.h"

int main() {
    . . .
```

Note that all three header files shown above use a `#ifndef/#define/#endif` construct. Which, if any, of those are necessary in order for this program to compile?

1) in `Name.h`
2) in `StudentID.h`
3) in `RoomAssignment.h`
4) All of them
5) 1 and 2 only
6) 1 and 3 only
7) 2 and 3 only
8) None are needed.
9) None of these

For questions 14 through 17, consider a program consisting of five functions that make calls as shown in the following partial structure chart:



The function implementations are organized into three `cpp` files, so that `main()` is in `main.cpp`, `Dogbert()` and `Boss()` are in `Dogbert.cpp`, and `Wally()` and `Alice()` are in `Wally.cpp`. There are also two header files, `Dogbert.h` and `Wally.h`. The following questions consider the contents of those header files.

One goal is to organize the header files so that each cpp file to be separately compilable. Another is to not make too many declarations (including function prototypes) available in scopes where they are not needed, but also that each declaration can appear only once.

14. Which functions' prototypes should go in `Dogbert.h`?

1) `Dogbert()`
2) `Boss()`
3) `Wally()`
4) `Alice()`
5) All of them
6) 1 and 2 only
7) 1 and 3 only
8) 1 and 4 only
9) No prototypes should go there.

15. Which functions' prototypes should go in `Wally.h`?

1) `Dogbert()`
2) `Boss()`
3) `Wally()`
4) `Alice()`
5) All of them
6) 1 and 2 only
7) 3 and 4 only
8) 1, 3 and 4 only
9) No prototypes should go there.

16. Which `cpp` files would need an include directive for the header file `Dogbert.h`?

1) `main.cpp`
2) `Dogbert.cpp`
3) `Wally.cpp`
4) All of them
5) 1 and 2 only
6) 1 and 3 only
7) 2 and 3 only
8) None of these

17. Which `cpp` files would need an include directive for the header file `Wally.h`?

1) `main.cpp`
2) `Dogbert.cpp`
3) `Wally.cpp`
4) All of them
5) 1 and 2 only
6) 1 and 3 only
7) 2 and 3 only
8) None of these

---

18. Which of the following are advantages that may result from using separate compilation instead of placing all the source code for a program in a single file?

1) Less total source code needs to be written.
2) Portions of the source code are potentially easier to re-use in other programs.
3) Re-compiling the source code after modifications may take less time.
4) All of them.
5) 1 and 2 only
6) 1 and 3 only
7) 2 and 3 only
8) None of these

---

19. Suppose you have implemented a class and wish to make it available, as is, to other developers. Which of the following common software engineering goals could be promoted by placing the class declaration in a header file and its implementation in a separate source file?

1) Encapsulation, because the data and operations are bundled together.
2) Information hiding, because you could distribute the header file with an object code file obtained by compiling the implementation file.
3) Both 1 and 2
4) None of these

---

20. What is the syntactic purpose of a header file associated with a particular source file?

1) To "publish" the declarations of all of the entities defined in the source file.
2) To "publish" the declarations of those entities defined in the source file which need to be used in another compilation unit.
3) To justify the inclusion of `#ifndef` and `#endif` in the C++ language.
4) None of these