CS 246 S2009 Midterm

1.

1. To prevent double inclusion - so classes aren't defined twice.
2. A
3. A
4. So the variable is not modified in the function.

5.

```
std::vector<Face> faces = { TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN, JACK,
QUEEN, KING, ACE };
std::vector<Suits> suits = { DIAMOND, CLUB, HEART, SPADE };

int i = 0;
for (auto face : faces) {
        for (auto suit : suits) {
                Card card {face, suit};
                deck[i] = card;
                ++i;
        }
}
```

2.

1. d
2. b
3. e
4. b

5.

```
delete n1;
delete p1;
delete n2->person_ptr;
delete n2;
delete n3->person_ptr;
delete n3;
}
```

6. b

7.

```
MyClass* obj = new MyClass();
MyClass *obj2 = obj;

delete obj;
obj = nullptr;
```

8. Line 3 (delete ptr) will cause an error, because b was not allocated on the heap, and delete can only be called on pointers to heap-allocated objects.

3.

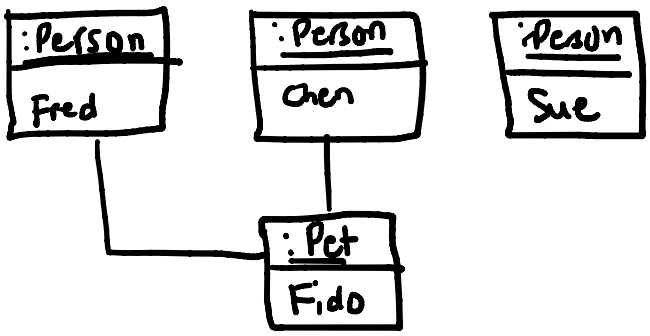
1. b
2. b
3. a
4. a
5. a
6. k
7. c
8. c
9. a
10. f
11. So it's called from std and not called from the object.

12.

```
double re, im;
char temp;
std::cin >> re >> temp >> im;

c.realIs(re);
c.imaginaryIs(im);
return std::cin;
```

4. 1.



2. This does not conform to the class diagram. According to the class diagram, each Pet can only be associated with one Person, but in the object model, the Pet Fido is associated with both Fred and Chen.

5.

1.
```
Name::~Name() {
	delete[] middle_;
}
```

2.
```
Name::Name(const Name &name) :  first_(name.first_), last_(name.last_), middle_(new
string[name.numMiddle_]), numMiddle_(name.numMiddle_) {
	for (int i = 0; i < numMiddle_; ++i) {
		middle_[i] = name.numMiddle_[i];
	}
}
```

3. Use copy constructor to copy fields, swap data with current object, return *this.

4. We can use the pointer to implementation idiom - create a class for all internal fields and functions, and just have a pointer to the implementation class.

6. 1. a
2. e
3. a
4. b
5. a

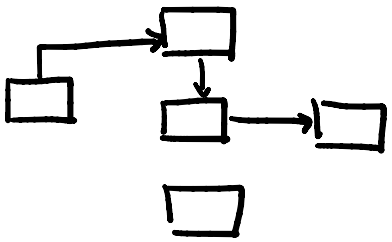
6.
```
#include <iostream>
#include <vector>

class CourseOffering {
        std::string term;
        Course* course;
        Professor* professor;
        std::vector<Student*> students;
public:
        CourseOffering(Course*, std::string);
        void enroll(Student*);
};
```

7.
Course:
        string getInstructorName(string term)
CourseOffering:
        string getInstructorName()
Professor:
        string getName()

Client module code:
        string instructorName = course.getInstructorName(string term);

8.



7. 1. a) 4 b) 6 c) 3 d) 2

2.
deck.o : deck.cc deck.h card.h
\t g++ -c deck.cc

3. c
The error was caught and reported - the code had good error handling