Review Questions II

The following review questions are similar to the kinds of questions you will be expected to answer on Exam II (Nov. 23), which will cover LCR, chs. 1–7 and 10, and TCS, chs. 1–18. Make sure you do your best to answer each question before you look at the Key; that will help you to learn. If you have any questions about the answers, please ask them in class or ask your Teaching Assistants. The actual exam will be about 16 questions.

1. (5 points)

Write a C++ `main()` function that reads `double` numbers from a file called "`in.dat`" until it reaches the end-of-file, and prints the numbers on `cout` left-justified in a field of width 6 with three digits after the decimal point. Your program should not use vectors or arrays (which it does not need). Make sure to show any required `#includes`.

2. (5 points)

What is the effect of the following statements?

```
int x, a = 11, b = 7;
x = a - b;
a = a - x;
b = a + x;
assert (b == 0);
```

3. (5 points)

State in your own words the difference between *reactive* (or *direct*) *control* and *behavior-based control*.

4. (5 points)

For the following questions, tell (1) what the function does with the argument, (2) what can be done to the argument within the function, and (3) how the original argument is affected by the action of the function.

(a) When an argument is passed by **value** to a function [e.g., `f(int v)`], what happens?

(b) When a r**eference variable** is passed to a function [e.g., `f(int& v)`], what happens?

(c) When a **constant reference** variable is passed to a function [e.g., `f(const int& v)`], what happens?

5. (5 points)

Define an enumeration types `Suit` and `Rank` for suits of cards (`CLUBS`, `DIAMONDS`, `HEARTS`, `SPADES`) and their ranks (`ACE`, `TWO`, …, `KING`). The `Rank` enumeration should be defined so that `ACE` has the numerical value 1 and the others follow consecutively (`TWO` = 2, etc.).

6. (5 points)

Who can see member variables and functions that are declared:

(a) `public`

(b) `private`

(c) `protected`

7. (5 points)

In the space provided below, indicate what gets printed from the following program?

```
#include <iostream>
using namespace std;
void trace(int, int *);

int main()
{
  int x = 43, y = 15, *ptr = &x, *q = &y;

  trace(x, q);
  cout << "In main:  " << *ptr << "  " << *q << endl;
}

void trace(int x, int *q)
{
  int y;

  y = 2;

  *q += y;
```

```
  y = x / 10;
  cout << "In trace:  " << y << "  " << *q << endl;
}
```

8. (5 points)

You have the following code fragment. What is printed with the indicated input values?

```
int x;
cin >> x;
switch(x)  {
    case 40:  x %= 6;
    case 60:  x--;  x--;  break;
    case 80:  x += 586;
    default:  if(x != 0)  x = -x;
}
cout << x << endl;
```

a) input: 40 __ _____

b) input: 80_____

9. (5 points)

Complete function `read_vec(N)` to read values into an integer vector of size `N` and to return it. Stop when either the vector is full or you reach eof. You may assume valid data.

```
vector<int> read_vec (const int N) {









  return v;
}
```

10. (5 points)

What does the word `virtual` mean in front of a function declaration?

11. (5 points)

In the Tic Tac Toe program discussed in ch. 10 of *Learning Computing with Robots* we discussed a function with the following declaration:

```
int lookahead (Board board, char player, bool MAX, int level);
```

Based on your understanding of the program, describe (1) the purpose of the function, (2) the meaning of each parameter, and (3) the meaning of the returned value.

12. (5 points)

Complete the following definition of a member function `sub()`, which subtracts two complex numbers. That is, `X.sub(Y)` should return the complex number obtained by subtracting the complex number `Y` from the complex number `X`. (Recall that the real part of the difference is the difference of the real parts, and the imaginary part of the difference is the difference of the imaginary parts.)

```
class complex {
public:
  double real, imag;
  complex sub (complex Y) {
    complex dif;









    return dif;
  } // end of sub
};
```

13. (5 points)

Write a definition to overload the "<<" operator to print out a complex number (as defined in question #12) . If X is the real part and Y is the imaginary part, it should print it in the form "`X + Yi`" if Y is nonnegative, and as "`X - Yi`" if Y is negative (that is, if X = 2 and Y = –1 it should print "`2 - 1i`". Treat the operator as a standalone function.

14. (5 points)

Suppose that we extend the declaration of complex in question #12 to include prototypes for three constructors:

```
complex ();               // constructor a
complex (double);         // constructor b
complex (double,double); // constructor c
```

Implement these constructors as described below:

a. Implement `complex()` to create a complex number with 0 real and imaginary parts.

b. Implement `complex(X)` to create a complex number with real part `X` and imaginary part 0.

c. Implement `complex(X,Y)` to create a complex number with real part `X` and imaginary part `Y`.

15. (5 points)

You have vectors `U` and `V` declared as:

```
vector <int> U (10); // create vector with 10 elements, all 0
vector <int> V (10, -1); // create vector with 10 elem., all -1
```

Write code to copy `V` to `U`. Hint: One line of code is all that is needed.

16. (5 points)

Explain what is wrong with the following code segment? Look carefully!

```
int X = new int;
*X = 1;
*X = 0;
delete X;
```

17. (5 points)

Given the following structure definitions, write an output command to print the `suit` of the last card in `deck`. (Note that decks need not contain 52 cards.)

```
struct Card { int suit, rank; };
struct Deck {
  vector<Card> cards;
```

```
};
Deck deck;
// deck initialization would be here
// your answer:
```

18. (5 points)

Suppose that `Card` and `Deck` are defined as in question #17, and suppose that you are given a `bool` function `greater(c1, c2)` that tests if card `c1` is greater than card `c2`. Complete the following definition of a `bool` function `ordered(d)` that returns `true` if deck `d` is sorted in ascending order and `false` otherwise. (Note that `d` may have less than 52 cards.)

```
bool ordered (const Deck& d) {




}
```

19. (5 points)

What does *inheritance* refer to in the context of object-oriented programming?

20. (5 points)

Given the following definitions, write code to give Joe a 15% raise and his supervisor a 5% raise:

```
class employee {
public:
  string name;
  double salary;
  employee* supervisor;
};
employee* Joe;
/* omitted code to initialize the employee records,
   including Joe's */
// put your code here:
```

21. (5 points)

Define a subclass of `employee` (see question #20) called `manager` that has an additional public member variable, `subordinates`, which is a vector of pointers to `employee` records.

22. (5 points)

State the *Information Hiding Principle*.

23. (5 points)

Consider the following function `alike()`, which returns `true` if two integer vectors are alike, and `false` otherwise. The vectors are considered different if they have different lengths.

```
bool alike(const vector<int>& A, const vector<int>& B){
   bool same = true; // assume vectors are alike
   if (A.size() != B.size()) same = false;
   for(int i = 0; i < A.size() && same; i++)
     if(A[i] != B[i])  same = false;
   return same;
}
```

Modify this function so that it will work on vectors of any type (e.g., vectors of `int`s, vectors of `double`s, vectors of `string`s, and so on).

24. (5 points)

Given the following declarations:

```
struct Node {
    int cargo;
    Node* next;
}
Node* data;
```

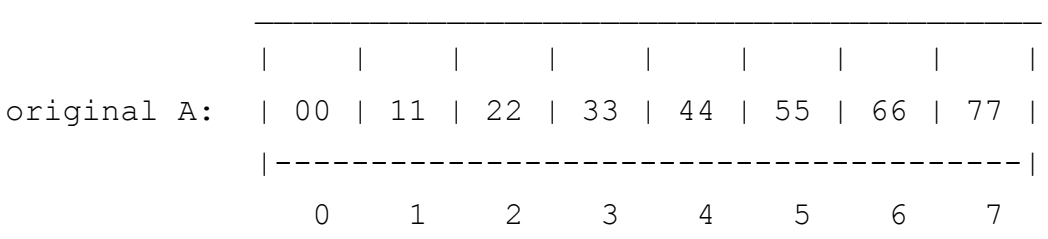
Write code to remove the second element of the linked list `data` and to return it to free storage. (Assume that `data` points to a linked list with at least two `Node`s.)

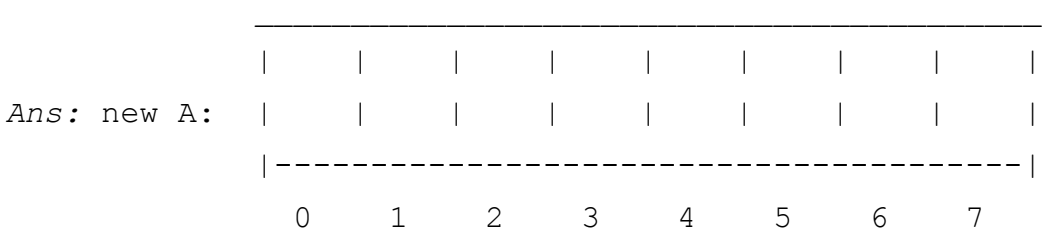
25. (5 points)

Fill in the new array values after the code segment is executed. Leave no blank array elements.

```
const int N = 8;
int i, A[N] = {0, 11, 22, 33, 44, 55, 66, 77};

for (i = 1; i < N-1; i++)      // look carefully!
  A[i] = A[i+1];
```





26. (5 points)

Suppose you are doing a *bisection search* for the number `2000` in a 64-element vector that has been initialized to 0, 100, 200, ..., 6300:

```
vector<int> V (64);
for (int i=0; i<64; i++) V[i] = i*100;
```

Write down the low and high index for each stage of the bisection search until it finds the number `2000`. To give you a start, I have written down the first few:

```
 0   63
 0   30
16   30
```

27. (5 points)

Given the following definition of `Complex`, define appropriate accessor functions to set and retrieve the private member variables. (Don't confuse this definition of `Complex` with the `complex` class used in other questions.)

```
class Complex {
private:
  double imag, real;
public:
  // you fill this in:









} // end of Complex
```