Part I: 11 Multiple choice questions (2 points each)

Answer all of the following questions. READ EACH QUESTION CAREFULLY. Fill the correct bubble on your mark-sense sheet. Each correct question is worth 2 points. Choose the one BEST answer for each question. Assume that all given C++ code is syntactically correct unless a possibility to the contrary is suggested in the question.

In code fragments, YOU SHOULD ASSUME THAT ANY NECESSARY HEADER FILES HAVE BEEN INCLUDED. For example, if a program does input or output, you should assume that header files for the appropriate stream library have been #included, and that “using namespace std;” has been provided, even if it is not shown in a question.

Remember not to devote too much time to any single question, and good luck!

1. Consider the following program:

```
#include <iostream>
#include <string>
using namespace std;

int main () {
   string str = "This*is^a.45min test.";
   int i;

   for (i = 0; i  < str.length( ); i++) {
      if (ispunct(str[i]))
         str[i] = ’ ’;          // a blank
      str[i] = tolower (str[i]);
   }

   cout << str;
}
```

What is printed by the last line of the code ?

A. this*is^a.45min test.

B. thisisa45mintest

C. **this is a 45min test**

D. this is a 45min test.

2. What is *cout* ?

A. It is a function

B. It is an operator

C. It is a class

D. **It is an object (class instance)**

E. It is a reserved word (C++ keyword)

3. Given below are some statements about the default (0-argument) constructor:

I. Its return type is the type of the class
II. It has no return type
III. The programmer can define it, but the C++ language doesn't require this
IV. The programmer must define it
V. It is always defined by C++ if it isn't provided by the programmer
VI. It is sometimes, but not always, defined by C++ if it isn't provided by the programmer

Which of these statements are true?

A. I, III and V only
B. I, II and VI only
C. II and IV only
D. II, III and V only
E. **II, III and VI only**

4. Which of the following functions will correctly return *true* if its argument is an odd integer ?

I.
```
bool IsOdd (int x) {
   return (x % 2 == 1);
}
```

II.
```
bool IsOdd (int x) {
   return (x / 2 == 1);
}
```

III.
```
bool IsOdd (int x) {
   if (x % 2 == 1)
      return true;
   else
      return false;
}
```

A. II only
B. I and II only
C. **I and III only**
D. II and III only
E. I, II and III

5. When an ADT is implemented as a C++ class, which of the following should normally be true ?

A. Member functions are private, member variables are public
B. **Member functions are public, member variables are private**
C. Member functions as well as member variables are private
D. Member functions as well as member variables are public

6. Given below are three implementations of the swap function:

I.

```
void swap (int a, int b) {
   int temp;
   temp = a;
   a = b;
   b = temp;
}

int  main () {
   int  i = 0, j = 1;
   swap (i, j);
}
```

II.

```
void swap (int &a, int &b) {
   int temp;
   temp = a;
   a = b;
   b = temp;
}

int main  () {
   int i = 0, j = 1;
   swap (i, j);
}
```

III.

```
void swap (int *a, int *b) {
   int *temp;
   temp = a;
   a = b;
   b = temp;
}

int main ()  {
   int i = 0, j = 1;
   swap (&i, &j);
}
```

Which of these would actually swap the contents of the two integers *i* and *j* ?

A. I only
B. **II only**
C. III only
D. I and II only
E. II and III only

7. What is printed by the following program ?

```
void func (int *b) {
   *b = 1;
}

int main () {
   int *a;
   int n;
   a = &n;
   *a = 0;
   func (a);
   cout << *a << endl;
}
```

A 0

B. **1**

C. The address of b

D. The address of a

E. The address of n

8. Consider the following class:

```
class FooBar {
   public:
      void f1 (string s);
      void f2 (const string &s);
      void f3 (string s) const;
   private:
      string str;
};
```

Which of the three member functions could legally alter member variable *str* ?

A. The function f1 only

B. The function f2 only

C. The function f3 only

D. **Two of them**

E. All three of them

9. Consider this piece of code:

```
void mysterious(int i, int &k) {
   i = 1;
   k = 2;
}

int main () {
   int x = 0;
   mysterious (x, x);
   cout << x << endl;
   return 0;
}
```

What is the value of *x* that gets printed by the *main* ?

A. 0

B. 1

C. **2**

D. None of these

10. Consider the following statements:

```
int *p;
int i, k;
i = 142;
k = i;
p = &i;
```

Which of the following statements changes the value of *i* to 143 ?

A. k = 143;

B. *k = 143;

C. p = 143;

D. ***p = 143;**

E. More than one of the above

11. In the following function:

```
int f (int n) {
   int v;
   v = 2*n+1;
   return v;
}
```

What is the storage class of variable v?

A. static

B. dynamic

C. contextual

D. **automatic**

E. inline

Part II: Short Answers (7 points)

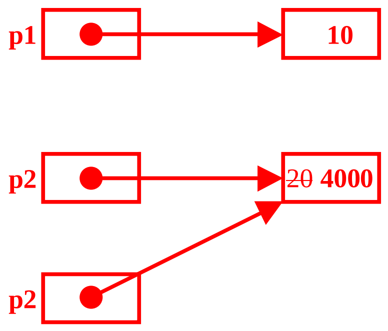
12. **(3 points) In the C++ language, there's a precise distinction made between a *declaration* of a function and a *definition* of a function. *Briefly* describe the distinction.**

In C++, a function *declaration* gives the function name, result type, and number and types of parameters. This is the function prototype, and it may be repeated or included in as many separate source files as necessary.

A function *definition* is the actual function, containing the code that makes up the body of the function. It must appear exactly once in the program.

13. **(4 points) Draw a boxes and arrow diagram showing the result of executing the following statements.**

```
int *p1, *p2, *p3;
p1 = new int;
*p1 = 10;
p2 = new int;
*p2 = 20;
p3 = p2;
*p3 = *p3**p2**p1;
```



Part III: Programming Question (12 points)

14. Here is the specification of class WordList from the sample solution to Homework 2. (Even if you had a different specification in your own assignment, you *must* use this specification in your answers to this question.)

```
const int MAX_WORDS = 300;

struct WordInfo {       // information about a single word
   string word;         // the word itself
   int frequency;       // number of times the word has appeared so far
};

class WordList {
   public:
   // construct empty WordList
      WordList ();
   // add the word to the WordList if not already present, else increase its frequency by 1
      void add_word (string word);
   // sort the WordList by frequency, using alphabetical order to break ties of frequency
      void sort_list ();

   private:
      WordInfo entry[MAX_WORDS];   // <word,frequency> pairs are stored in
      int nWords;                  //       entry[0..nWords-1]
};
```

For this problem, add to class WordList an integer-valued member function *nWordsWithFrequency(n)* that returns the number of words in the WordList whose frequency is n. That is, the function should count how many different words in the WordList have a frequency = n, and return that count.

(a) (3 points) Write an appropriate function prototype for *nWordsWithFrequency* that would be included in the WordList class in *WordList.h*. Be sure to include appropriate parameters and return type (if any) and an appropriate heading comment for the function.

```
// = # of words in this WordList that have a frequency = n
int nWordsWithFrequency(int n);
```

(b) (7 points) Give a correct implementation of *nWordsWithFrequency*, which would be included in the implementation file, *WordList.cpp*.

```
// = # of words in this WordList that have a frequency = n
int WordList::nWordsWithFrequency(int n) {
  int nMatch = 0;          // # words encountered with frequency = n

  for (int i = 0; i < nWords; i++)
    if (entry[i].frequency == n)
      nMatch++;

  return nMatch;
}
```

(c) (2 points) Suppose we have a main program that includes a *WordList* w. Fill in the blank line below with an appropriate call to *nWordsWithFrequency* to store in nUnique the number of words that appear in WordList w with a frequency of 1.

```
int main () {
  WordList w;        // list of <word,frequency> pairs
  int nUnique;       // number of words with frequency = 1

  // code to read words and count them using WordList w omitted
  …

  // store the number of unique words in nUnique:

  nUnique = w. nWordsWithFrequency(1) ;       // fill in this blank
  …

  return 0;
}
```