

1.) [8 points] Correct the error(s) in each segment of code below. You will get points off for changing code that was not in error.

a.)

```
int x;
cin >> x;
string words[x];
words[0] = "Gandalf"
```

Array size must be a constant.

Change to vector:

`vector<string> words(x);`

b.)

```
int y;
cin >> y;
if 0<y<10
    cout << "y is between
           0 and 10";
```

Incorrect if statement:

`if (0<y && y<10)`

c.)

```
double investment = 2000.0;
double return = 1.016 * investment;
cout<< setprecision(2) << fixed;
cout<< "The return on your investment is $" << return;
```

Cannot use return as variable, it is a reserved word. Change return to my_return

d.)

```
char c;
cin >> c;
if (c=="f" && c=="F")
    cout << "You typed an upper or lower-case F.";
```

Will never run because can't be both f and F. Also chars have single quotes.

`if (c=='f' || c=='F')`

2.) [6 points] Briefly comment what each line of code below does.

```
int a[4]={10,20,30,40}; //Create an array a with values 10,20,30,40.
int* p; int* q; //Create integer pointers p and q.
p = new int; //Allocate new integer memory block to p.
*p = 55; //Assign value 55 to block p points to.
q = a; //Set q pointing to first element of array a.
*q = *p; //Assign value q points to (a[0]) same as p (55).
delete p; //Free up the integer block p points to.
```

3.) [12 points] Write the output of each of the code segments below.

a.)

```
string s = "gandalf";
s.insert(3, "SAM");
cout << s << "\n";
s.erase(2,4);
cout << s << "\n";
s = s.substr(2,4);
cout << s;
```

ganSAMdalf

gadalf

dalf

b.)

```
int a=4;
int j=1;
do {
    cout << j << "+";
    j = j + 2;
    if (j>a)
        j--;
} while (j<8);
```

1+3+4+5+6+7+

c.)

```
cout << setprecision(2);
for (int i=3; i>0; i--) {
    cout << i/3.0 << " ";
}
```

1 0.67 0.33

d.)

```
int a[4] = {3,4,-2,6};
for (int i=3; i>0; i--)
    a[i] += a[i-1];
for (int i=0; i<4; i++)
    cout << a[i] << " ";
```

3 7 2 4

4.) [9 points] Write the output of the program below. Track your variables in a walkthrough table as we did in lecture.

<pre>#include <iostream> using namespace std; int* fun(int* b, int* &c) { b--; *c = (*b)%2; c = c + 2; *c = *c + 2; cout << "b=" << *b << " c=" << *c << "\n"; return b + (*b); }</pre>	<pre>int main() { int a[5] = {3,4,-1,1,3}; int* p = a; int* q; while (p != &a[4]) { q = fun(&a[1], p); cout << "p=" << *p << " q=" << *q << "\n"; } for (int i=0; i<5; i++) cout << a[i] << " "; return 0; }</pre>
--	---

```
b=1 c=1
p=1 q=4
b=1 c=5
p=5 q=4
1 4 1 1 5
```

5.) [10 points] A *palindrome* is a word that is the same backwards and forwards. For example, the strings below are all palindromes:

ODO dood racecar H*q*H 123321

Write a boolean function `isPalindrome` that returns true if a given string is a palindrome and false if the string is not a palindrome.

```
bool isPalindrome (string word) {
    int size = word.length();
    for (int i=0; i<size/2; i++)
        if (word[i] != word[size-i-1])
            return false;
    return true;
}
```

6.) [10 points] Write a function that returns a vector of integers that contains alternating elements of two integer vectors called `a` and `b`. Begin with the first element of `a`, then alternate until one of the vectors is out of elements. After that point, simply include the remaining elements of the longer vector.

```
vector<int> merge (const vector<int>& a,
                  const vector<int>& b) {
    vector<int> c;
    int i = 0;
    while (i<a.size() || i<b.size()) {
        if (i<a.size())
            c.push_back(a[i]);
        if (i<b.size())
            c.push_back(b[i]);
        i++;
    }
    return c;
}
```

7.) [15 points] Write a full program (starting from #include) that reads in a text file specified by the user and the name of a destination output file. The program should output the words of the input file in the reverse order in which they appeared. You may assume the input consists of text separated by spaces. You do not need to worry about punctuation.

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

int main() {
    string infile, outfile;
    cout << "Enter input file name: ";
    cin >> infile;
    cout << "Enter output file name: ";
    cin >> outfile;
    ifstream fin;    //Create input file stream.
    fin.open(infile.c_str());
    ofstream fout;   //Create output file stream.
    fout.open(outfile.c_str());
    string temp;
    vector<string> words;
    while (fin >> temp)    //Read in all words in input file.
        words.push_back(temp);
    for (int i=words.size()-1; i>=0; i--)    //Reverse order.
        fout << words[i] << " ";
    fin.close();    //Remember to close your streams!
    fout.close();
    return 0;
}
```

8.) [15 points] A character double occurs when the same character occurs twice in a row, such as the "bb" in "hobbbit". Note the character could be a symbol, such as "++" or "!!". For example, the words below all contain character doubles.

good hello C++ HOBBIT heehee

Write a full program (starting from #include) that counts the number of character doubles in a phrase entered by the user. You do not have to worry about detecting upper vs. lower-case letters, nor do you have to handle the case when a character occurs three times or more in a row.

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

int main() {
    string phrase;
    cout << "Enter phrase: ";
    getline(cin, phrase);
    int count = 0;    //Counts number of character doubles.
    for (int i=0; i<=phrase.length()-2; i++)
        if (phrase[i] == phrase[i+1])
            count++;
    cout<<"There are "<<count<<" character doubles.\n";
    return 0;
}    //Note: This exact problem was on Fall 2007 Exam 1.
```

9.) [15 points] Write the declaration and definitions for a Point2D class that stores the x and y coordinates of a two-dimensional point. The constructor for the Point2D class should take the x and y coordinates as input. The distance_from function should calculate the distance between two points, taking one Point2D as an implicit parameter and the other as an explicit parameter. Recall the distance between two points (x_1, y_1) and (x_2, y_2) is given by

$$\text{distance} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

You may use functions in the <cmath> library. We will compare two Point2Ds based solely on their x-coordinates. So point1 < point2 if the x-coordinate of point1 is less than the x-coordinate of point2.

An example of code using the Point2D class is shown below. Your Point2D class should define all functions necessary to run the code below.

```
Point2D p1 (3, -4.2);
Point2D p2 (10.2, 5.8);
cout << "The coordinates of p1 are x=" << p1.get_x()
      << " and y=" << p1.get_y() << ".\n";
cout << "The distance between p1 and p2 is "
      << p1.distance_from(p2) << ".\n";
if (p1 < p2)
    cout << "Point p1 is to the left of p2.";
else
    cout << "Point p1 is to the right of p2.";
```

```
class Point2D {
public:
    Point2D(double x0, double y0);
    double get_x() const;
    double get_y() const;
    double distance_from(Point2D p2) const;
    bool operator< (Point2D p2) const;
private:
    double x;
    double y;
};
```

```
Point2D::Point2D(double x0, double y0) {
    x = x0;
    y = y0;
}
```

```
double Point2D::get_x() const {
    return x;
}
```

```
double Point2D::get_y() const {
    return y;
}
```

```
double Point2D::distance_from(Point2D p2) const {
    return sqrt( pow(x-p2.x,2) + pow(y-p2.y,2) );
}
```

```
bool Point2D::operator< (Point2D p2) const {
    return x < p2.x;
}
```