

Final Exam, PIC 10A, Spring 2006

Sunday, June 11th 2006

Name: _____

Signature: _____

ID Number: _____

Discussion Section: _____

Problem	1	2	3	4	5	6	7	8	Total
Score									

- No use may be made of books, notes or electronic devices.
- Use only this exam packet and a non-red pen or pencil.
- Problems are not necessarily in order of difficulty! Do easy problem first.
- If you need more space, make a note and continue on the back of previous page.
- If something is unclear to you, raise your hand and I or a TA will come to you

Problem 1 (20 points) Find the correct answer and circle it from the following choices.

1) Where is a variable declared to make it local to a function?

- a) in the function prototype
- b) in the main function
- c) in the function definition
- d) before the main

ANS: C

2) What operator is used to access a class through a pointer?

- a) ->
- b) >>
- c) *
- d) &

ANS: A

3) What will strcmp("Astring", "Astring") return?

- a) a positive value
- b) a negative value
- c) zero
- d) true

ANS: C

4) Given the code `int *p=new int(5);` What is the value of *p?

- a) the address of a dynamic integer variable
- b) 5
- c) the address of the first index variable of a dynamic array
- d) int

ANS: B

5) Which library you should include if you want to use predefined toupper(char) function?

- a) `#include <iostream>`
- b) `#include <fstream>`
- c) `#include <string>`
- d) `#include <cctype>`

ANS: D

6) Given the code

```
ifstream fin;  
fin.open("infile.txt");  
int onenumber;  
fin >> onenumber;
```

Which of the following statement is false?

- a) `fin` is declared to be an input stream for a file
- b) if `infile.txt` does exist, `fin.open("infile.txt")` will connect the input stream `fin` to the file named `infile.txt`

- c) read an integer from the file `infile.txt` into `onenumber`, if the file contains an integer
d) if `infile.txt` doesn't exist, `fin.open("infile.txt")` will create an empty file named `infile.txt`

ANS: D

7) The difference between `'\n'` and `"\n"` is

- a) `'\n'` is a value of type **char** and `"\n"` is a **string**
- b) produce the same effect, get a new line
- c) `"\n"` is not a char type and cannot be stored in a variable of type `char`
- d) both a) and c)

ANS: D

8) Under what circumstances may one use the following syntax for an array **a**

`a[] = {1, 2, 3, 4, 5}`

- a) anytime
- b) when declaring a
- c) when changing the values of the elements of a
- d) both (b) and (c)

ANS: B

9) What is the output of the following code

```
char ch;  
for(int k = 0; k < 3; k++)  
    cin >> ch;  
cout << ch;
```

If the input is

a b
c d

- a) a b c d
- b) just the return character
- c) c
- d) abcd

ANS: C

10) Consider the following code

```
int a[10], *p, *q;  
p = &a[3];  
q = p;  
q+=2;
```

what is stored in `q`

- a) the value of `a[3]`
- b) the address of `a[3]`
- c) the value of `a[5]`
- d) the address of `a[5]`

ANS: D

Problem 2 (12 points) what output will be produced by the following code, when embedded in a complete program?

```
a) int a = 5;
   while( a <= 160)
   {
       a *=2;
       cout << a << endl;
   }
   cout << "a is now " << a << endl;
```

10
20
40
80
160
320
a is now 320

```
b) int t = 2, s = 3, r = 5, n = 4;
   cout << t - s % 5 + r / t << endl;
   cout << double(t)/n << " " << t/n << endl;
   cout << r + n - s % n * t << endl;
```

1
0.5 0
3

Problem 3 (10 points) circle the mistakes and write the corrections on the margin. Make as few changes as possible. Here are four sample runs of what the program must do.

Integer? 0
0 is zero and not in 1..9.
sqrt(x) is about 0.

Integer? -1
-1 is neg. and not in 1..9.

Integer? 6
6 is pos. and composite.
Sqrt(x) is about 2.44949

Integer? 7
7 is pos. and prime.
sqrt(x) is about 2.64575

```
#include <iostream>

-----1) #include<cmath>
using namespace std;

void printSign(int x)
{
    if(x = 0) -----2) if(x == 0)
        cout << "zero";
    else if(x > 0) cout << "pos.";
    else /* x < 0 */      cout << "neg.";
}

void printMoreInfo(int x)
{
    if(1 <= x <= 9) -----3) 1 <= x && x <=9
    {
        switch(x)
        {
            case 2: case 3:
            case 5: case 7: cout << "prime"; -----4) break;
            default: cout << "composite";
        }
    }
    else cout << "not in 1..9";
}

int main(void)
{
    cout << "Integer? ";
    int x; cin >> x;

    cout << x << " is ";
    printSign(x);
    cout << " and " ;
    printMoreInfo(x);
    cout << ".\n";

    if(0 <= x) -----5) if(0 <= x)
        cout << "sqrt(x) is about " << sqrt(x) << ".\n";
    return(0);
}
```

Problem 4 (12 points) There are two parts for this problem.

- a) Write a function to compute the factorial of an integer n (also written as $n!$)
The function should take one parameter of type `int(n)` and return an argument of type `int(n!)`. $n!$ is computed as follows: if $n = 0$, $n! = 1$; otherwise, $n! = n * (n-1) * \dots * 1$. For example $4! = 4 * 3 * 2 * 1 = 24$.

```
int factorial(int n)
{
    int fac = 1;

    if( n < 0)
        cout << "your integer is invalid! ";
    else if( n == 0)
        return (1);
    else
    {
        for(int i = 1; i <= n; i++)
            fac *= i;
    }

    return (fac);
}
```

- b) use a switch statement to rewrite the following code.

```
int x;
cout << "input the value of x :";
cin >> x;

    if(x == 1)
        cout << " It's a busy day!" << endl;
    else if(x == 3 || x ==4)
        cout << "Good!" << endl;
    else
        cout << "All right!" << endl;
```

```
int x;
cout << "input the value of x: ";
cin >> x;
switch(x)
{
    case 1:
        cout << "It's a busy day!" << endl; break;
    case 3:
    case 4:
        cout << "Good!" << endl; break;
    default:
        cout << "All right!" << endl; break;
}
```

Problem 5 (12 Points)

a) Consider the following fragment. What is the output?

```
int balance = 40, child = 2;

if( balance > 50 && child < 2)
    cout << "tax = " << balance*0.30 << endl;
else if( balance > 50 && child > 2)
    cout << "tax = " << balance*0.20 << endl;
else if( balance > 30)
    cout << "tax = " << balance*0.15 << endl;
else
    cout << "tax = " << balance*0.10 << endl;
```

tax = 6

b) Consider the following type definition. Finish the main function to determine which box has bigger volume.

```
struct Box
{
    int length;
    int height;
    int width;
};

int main()
{
    Box box1, box2;

    int vol1 = box1.length * box1.height * box1.width;

    int vol2 = box2.length * box2.height * box2.width;

    if (vol1 < vol2)
        cout << "Box2 is bigger.";
    else if (vol1 > vol2)
        cout << "Box1 is bigger.";
    else
        cout << "Box1 is as big as Box2.";

    cout << endl;
}
```

Problem 6 (10 points) The C++ standard library function

int strlen(const char *s)

Determines the length of the string s, the number of characters preceding the terminating null character is returned.

Write your own two versions of function strlen(). The first version should use array subscripts and subscript arithmetic, and the second version should use pointers and pointer arithmetic.

a)

```
int strlen(const char *s)
{
    unsigned long t = 0;
    int i = 0;
    while( s[i] != '\0' )
    {
        t++;
        i++;
    }
    return t;
}
```

b)

```
int strlen(const char *s)
{
    unsigned long t = 0;
    const char* p = s;
    while( *p != '\0' )
    {
        t++;
        p++;
    }
    return t;
}
```

Problem 7 (10 points) String

Use getline function to read up to 80 characters from user input. Write a program to find out how many vowels (a/A, e/E, i/I, o/O, u/U) in this sentence?

A sample output will be like the following

```
Please input your sentence: I Love You!
Input = I Love You!
Length of Input = 11
There are totally 5 vowels in this sentence.
```

```
#include <iostream>
#include <cstring>
#include <cctype>
using namespace std;

int main()
{
    char input[80];
    int length, count = 0;

    cout << "please input your sentence: ";
    cin.getline(input, 80);
    length = strlen(input);

    for(int i=0; i < length; i++)
    {
        switch(toupper(input[i]))
        {
            case 'A':
            case 'E':
            case 'I':
            case 'O':
            case 'U':
                count++;
                break;
            default:
                break;
        }
    }

    cout << "Input = " << input << endl;
    cout << "Length is = " << length << endl;
    cout << "there are totally " << count
         << " vowels in this sentence " << endl;
    return 0;
}
```

Problem 8 (14 points)class Point represents a single point on 2-D plane. This class has 2 private member variables, which are the corresponding x coordinate and y coordinate. It has 3 constructors to initialize its member variables. It also has 2 access and 2 mutation member functions. Member function distToOrigin() is to compute the distance between current point to origin(0,0). Member function distToAnotherPoint(Point &otherP) is to compute the distance between current point and point otherP. Member function print() is to print out the point in the format of Point(2,3), if xValue is 2 and yValue is 3.

a) Complete the definition of class Point by implementing those four member functions. (write your code in the white space)

- 1) Point::Point(Point &givenPoint)
- 2) double Point::distToOrigin()
- 3) double Point::distToAnotherPoint(Point &otherP)
- 4) void Point::print()

Example: For p1(2,3) and p2(4,5), the distance is $\sqrt{(p1.xValue-p2.xValue)^2 + (p1.yValue-p2.yValue)^2} = \sqrt{(2-4)^2 + (3-5)^2}$.

```
#include <iostream>
#include <cmath>
#include <cstdlib>
#include <ctime>
using namespace std;

class Point
{
public:
    Point() {xValue=0.0; yValue=0.0; } // default constructor
    Point(double x, double y) { xValue=x; yValue=y; }
    Point(Point &givenPoint)
    {
        xValue = givenPoint.getXcoordinate();
        yValue = givenPoint.getYcoordinate();
    }
    double getXcoordinate() { return (xValue); } // access methods
    double getYcoordinate() { return (yValue); }
    void setXcoordinate(double x) { xValue = x; } // set methods
    void setYcoordinate(double y) { yValue = y; }
    double distToOrigin()
    {
        double d = sqrt(xValue*xValue + yValue*yValue);
        return d;
    }
    double distToAnotherPoint(Point &otherP)
    {
        double dx = xValue- otherP.getXcoordinate();
        double dy = yValue- otherP.getYcoordinate();
        return (sqrt(dx*dx+dy*dy));
    }
}
```

```

        void print()
        {
            cout << "Point(" << xValue << "," << yValue << ")";
        }
private:
    double xValue;
    double yValue;
}

```

b) use the class Point to complete the main functions. (write your code in the white space)

Given 20 points on 2-D plane, find the maximum and minimum distance among those points. 20 points are given in the form of 2 arrays. The first array stores x coordinates, the second array stores y coordinates. You need to construct 20 points into a Point array and write your program based on this Point array.

The sample output will be like the following,

Those 20 points are:

```

Point(1,2)   Point(9,2)   Point(3,2)   Point(9,3)   Point(9,7)
Point(0,6)   Point(6,4)   Point(3,3)   Point(5,6)   Point(5,9)
Point(3,4)   Point(0,3)   Point(6,9)   Point(5,8)   Point(5,7)
Point(5,1)   Point(9,9)   Point(1,5)   Point(3,6)   Point(5,4)

```

The maximum distance between any two points is 10.8167,

it's between Point(0,3) and Point(9,9);

the minimum distance between any two points is 1,

it's between Point(9,2) and Point(9,3).

```

void initialize(double x[], double y[], int size)
{
    srand(time(0));
    for(int i=0; i < size; i++)
    {
        x[i]=double(rand()%10);
        y[i]=double(rand()%10);
    }
    return;
}

int main()
{
    int n = 20;
    double x[n], y[n];
    //initialize maxDist and minDist
    double maxDist=0.0, minDist=10000000.0;

    // compute the distance between any two points v[i], v[j],
    // find the maximum distance and the minimum distance,
    // then print out them along with the corresponding points.
    // stores all points

    Point v[n];
}

```

```

// maxP1 and maxP2 are indexes of points between which lies the maximum
// distance, minP1 and minP2 are indexes of points between which lies
// the minimum distance
int maxP1 = 0, maxP2 = 0, minP1=0, minP2=0 ;

initialize(x, y, n);
// your code starts here
    for(int i = 0; i < n; i++)
        v[i]=Point(x[i], y[i]);

    for(i = 0; i < n-1; i++)
    {
        for(int j = i+1; j < n; j++)
        {
            double temp=v[i].distToAnotherPoint(v[j]);
            if(temp > maxDist)
            {
                maxDist=temp;
                maxP1=i;
                maxP2=j;
            }
            if(temp < minDist)
            {
                minDist=temp;
                minP1=i;
                minP2=j;
            }
        }
    }
    cout << "Those 20 points are:";

    for(i = 0; i < n; i++)
    {
        if (i%5==0) cout << endl;
        else cout << " ";
        v[i].print();
    }
    cout << endl << endl;
    // print result as in sample
    cout << "The maximum distance between any two points is "
        << maxDist << ", " << endl;
    cout << "it's between ";
    v[maxP1].print();
    cout << " and ";
    v[maxP2].print();
    cout << ";" << endl << endl;

    cout << "the minimum distance between any two points is "
        << minDist << ", " << endl;
    cout << "it's between ";
    v[minP1].print();
    cout << " and ";
    v[minP2].print();
    cout << "." << endl << endl;
    return 0;
}

```