

Name: _____

SSN: _____

Login: _____

CS 12 - Final June 17, 1998

Be sure to read each problem carefully and follow the directions. Points may be marked off if you do not follow the directions. For example, if the problem asks you to write a function, do not write an entire program. Please feel free to ask if you have any questions.

Problem 1	10	
Problem 2	10	
Problem 3	15	
Problem 4	12	
Problem 5	12	
Problem 6	12	
Problem 7	12	
Problem 8	8	
Problem 9	5	
Problem 10	6	
Problem 11	10	
Problem 12	10	
Problem 13	12	
Problem 14	12	
Problem 15	10	
Problem 16	10	
Problem 17	34	
TOTAL	200	

Name: _____

SSN: _____

Login: _____

1. (10 pts) Fill in the blank with the appropriate term.
 - (a) The _____ string function finds the first occurrence of a substring within a string.
 - (b) _____ control access allows non-member functions to access the member.
 - (c) A type template parameter is denoted by the keyword _____.
 - (d) Functions can be overloaded if their _____ differ.
 - (e) Using pass-by-_____ the actual argument may be affected.

2. (10 pts) Show the values of each element in the array **A** after the following code has been performed.

```
int A[5] = {6, 9, 11, 12, 1};
int *ip = A;

*ip = 20;
ip++;
*(ip+1) = 14;
*ip = *(ip+1) + 2;
ip += 2;
(*ip)++;
*(ip+1) += 4;
```

3. (15 pts) True/False (please write out “True” and “False”, do not just put a T or F).

- (a) A character array is a string.
- (b) Any problem that can be written recursively, can be written iteratively.
- (c) A base class object can be considered a derived class object.
- (d) Virtual functions are only used in conjunction with inheritance.
- (e) Anything that can be solved with operator overloading, can be solved with a regular function.
- (f) Operator overloading always requires the use of friends.
- (g) Classes are a form of polymorphism.
- (h) Anything that can be done using pass-by-reference, can be done using pass-by-value.
- (i) When declaring an object of a derived class type, the derived class constructor is always called first.
- (j) All string functions ensure that there will be a '\0' at the end of all strings they make changes to.
- (k) Multi-dimensional arrays are stored in row-major form in C++.
- (l) Dynamic allocation allows an array's size to be set at run-time.
- (m) After using the extraction operator (>>), the input stream is empty.

(n) The following two functions can be overloaded:

```
void func(int x, double y){ ... }  
void func(double y, int x){ ... }
```

(o) The following two functions can be overloaded:

```
void myfunc(int x, int y){ ... }  
int myfunc(int a, int b){ ... }
```

4. (12 pts) Define the following terms.

(a) Abstract Data Type

(b) Constructor

(c) Derived Class

(d) This Pointer

5. (12 pts) Given the following code, write in the letter of the corresponding correct expression type in the blank. An answer may be used more than once.

```

class account{
    private:
        char name[30];
        int accnum;
        double balance;
    public:
        ...
};
struct bank{
    account list[100];
    double total;
};
int *x;
int total;
bank BofA;
account A[10];

```

What are the types of the following expressions?

- | | | | |
|----------|----------------------|---|--------------------|
| _____ 1 | account | A | integer |
| _____ 2 | BofA | B | double |
| _____ 3 | x | C | character |
| _____ 4 | total | D | string |
| _____ 5 | A | E | array of integers |
| _____ 6 | *x | F | array of accounts |
| _____ 7 | bank | G | character pointer |
| _____ 8 | A[2] | H | integer pointer |
| _____ 9 | BofA.list | I | bank |
| _____ 10 | A[3].name | J | account |
| _____ 11 | BofA.list[1].balance | K | invalid expression |
| _____ 12 | *A[4].name | | |

6. (12 pts) Show what each variable contains after the following code has been executed. State what (if anything) the input stream still holds.

```
char buffer1[20];
char buffer2[20];
char buffer3[20];
char buffer4[20];
char ch;

cin.getline(buffer1, 20, ' ');

cin >> buffer2;

cin.get(ch);

cin.getline(buffer3, 4, '\n');

cin.get(buffer4, 20, '\n');
```

Assume that the user inputs the following:

```
‘‘Have a great summer.’’
```

7. (12 pts)
- (a) Explain two advantages of using classes.

(b) List two advantages of inheritance.

8. (8 pts) State what is wrong with the following code segments.

```
(a)  struct stats{
        int total;
        int count;
        double ave;
    };

    stats *Sp;
    Sp = new int;

    Sp.count = 4;
    Sp.total = 30;
```

```
(b)  class C
    {
        private:
            char name[30];
        public:
            void set(char *n)
            {
                name = n;
            }
    };

    void main()
    {
        char str[30];
        C myC;

        cout << "Enter name: ";
        cin >> str;
        myC.set(str);

        cout << myC.name << endl;
    }
```

(c)

```
template [type sometype]
void input(sometype list[])
{
    for(int i=0; i<size; i++){
        cout << 'Enter info: ';
        cin >> list[i];
    }
}
```

9. (5 pts) Translate the following code to use the iomanipulator flags. Assume **p1** and **p2** are doubles.

```
printf('The prices are: \n');
printf('%10.2f\n%10.2f\n', p1, p2);
```

10. (6 pts)

(a) Use *printf* to print a user defined string **str**.

(b) Now use *printf* to print the string one character at a time.

11. (10 pts) Write a recursive function that takes in one integer and returns the Fibonacci equivalent of that number. The Fibonacci of a number is defined as:

$$Fibonacci(n) = Fibonacci(n - 1) + Fibonacci(n - 2)$$

$$Fibonacci(1) = 0$$

$$Fibonacci(2) = 1$$

12. (10 pts) Overload the insertion operator for the **account** class from problem 5. Specify any lines that would need to be added to the class.

13. (12 pts) Assume a file called *prices* exists with a bunch of prices in it (in any format). Ask the user for an amount. Count the number of prices in the file that are less than or equal to user specified amount. Print the total number of occurrences and the user specified amount to the screen.

14. (12 pts) Write code to perform the following problem. Use the string functions wherever possible to get full credit. Ask the user to enter a string. Print out the first letter and last letter of each word that the user entered. For example if the user enters: "My name is Bob.", the following should be printed:

```
M y  
n e  
i s  
B b
```

15. (10 pts) Show code to simulate the definition of the class **myclass** without using inheritance.

```
class thisclass
{
    private:
        int x;
    protected:
        int var;
    public:
        thisclass(){ ... };
        void set(){ ... };
        void print(){ ... };
};

class anotherclass
{
    private:
        double num;
    protected:
        int list[10];
    public:
        anotherclass(){ ... };
        void setinfo(){ ... };
        void printinfo(){ ... };
};

class myclass : protected thisclass, public anotherclass
{
    private:
        int blah;
    public:
        myclass(){ ... };
        void setinfo(int x){ ... };
        void display(){ ... };
};
```

16. (10 pts) Assume that **C1** is a base class and **C2** is a derived class. Assume that the function **func** is a virtual function and that the function **func2** is not. Use the following code.

```
C1 Obj1;
C2 Obj2;

C1 *Ptr1;
C2 *Ptr2;
```

For each of the following code segments, state which class's function will be called. You have three answer options: *C1*, *C2*, and *error*.

- (a) `Obj1.func();`
- (b) `Obj2.func();`
- (c) `Ptr1 = &Obj1;`
`Ptr1->func();`
- (d) `Ptr1 = &Obj2;`
`Ptr1->func();`
- (e) `Ptr2 = &Obj1;`
`Ptr2->func();`
- (f) `Ptr2 = &Obj2;`
`Ptr2->func();`
- (g) `Obj1.func2();`
- (h) `Obj2.func2();`
- (i) `Ptr1 = &Obj1;`
`Ptr1->func2();`
- (j) `Ptr1 = &Obj2;`
`Ptr1->func2();`

17. (34 pts) Write code to do the following. You must write the code for all functions. You do not have to write a main. Reuse code wherever possible.

- (a) Write a class called **person** that maintains information about a person. It should have a data member to hold the person's name and a data member to hold the person's age. You should also have the following function members:

Set info Have a function that allows the information of a person to be set through parameters.

Print info Have a function that prints the information of a person nicely formatted.

- (b) Write a class called **family** that maintains information about a family. It should keep the number of people in the family. It should also keep information about each person in the family (do not waste space). You should have the following function members:

Constructor Write a basic constructor that does not take any parameters.

Destructor Perform any necessary clean up.

Set info This function should take one parameter, the number of people in the family. It should allow the user to set all the information for each person in the family.

Print info Print out the information for the family nicely formatted.

Copy Allow one to duplicate the family. Overload the assignment (=) operator.