

NAME

Solutions

1. Given the following data structures, on the next page write a segment of code to do each task.

```
class date
{
private:
    char place[100];
public:
    char month[10];
    int day;
    int year;
    void assignplace(char p[100])
    {
        strcpy(place, p);
    }
    void readplace(char p[100])
    {
        strcpy(p, place);
    }
}

class record
{
public:
    char name[100];
    char address[100];
    int paygrade;
    date hiredate;
    date DOB;
}

struct CompanyInfo
{
    char name[100];
    char address[10];
    date establish_date;
    record employee_records[100];
}

CompanyInfo JoesPizza;
```

Write a segment of code to do each task:

- a) Change the first letter of each employee's name to 'A'.

```
for (i = 0; i < 100; i++)  
    JoesPizza.employee_records[i].name[0] = 'A';
```

- b) Change the first letter of each employee's birth place to 'B'.

```
for (i = 0; i < 100; i++)  
    JoesPizza.employee_records[i].DOB.read_place(str),  
    str[0] = 'B';  
    JoesPizza.employee_records[i].DOB.assign_place(str);
```

- c) Change the company's established year to 2005.

```
JoesPizza.establish_data.year = 2005;
```

- d) Change employee # 37's hire date year to 2000.

```
JoesPizza.employee_records[37].hire_date.year = 2000;
```

- e) Print the record number for every employee whose name is the same as the company's name. Use strcmp( str1, str2 ) to compare 2 strings. It returns 0 when equal.

```
for (i = 0; i < 100; i++)  
    if ( strcmp ( JoesPizza.employee_records[i].  
                name, JoesPizza.name ) == 0 )  
        printf ( "%d", i );
```

2. Show the contents of each variable and the output of the following program

```

class MyClass
{
public:
    int a,
    int b;
    int c;
    MyClass(int x, int y, int z)
    {
        a = y;
        b = x;
        c = z*2;
        printf("A %d %d %d\n", a, b, c);
    }
}

MyClass MC1(1,2,3), MC2(4,5,6);
int a = 30, b = 40, x = 50;

void bill ( MyClass *MC)
{
    MC->b = b;
    MC2.c = MC1.a;
    MC->c = MC1.c + 1;
}

void main()
{
    int a = 60, c = 70, MC3 = 250;

    MC1.a = MC3;
    MC2.b = c;
    printf("B %d %d %d\n", MC1.a, MC1.b, MC1.c);
    bill(&MC1);
    printf("C %d %d %d\n", MC1.a, MC1.b, MC1.c);
    MC2.c = MC1.c;
    printf("D %d %d %d\n", MC2.a, MC2.b, MC2.c);
}

```

output

A 2 1 6  
A 5 4 12  
B 250 1 6  
C 250 40 7  
D 5 70 7

myclass (X, Y, Z)

<u>X</u>	<u>Y</u>	<u>Z</u>
1	2	3
4	5	6

file			MC2			Bill (*MC)			main		
MC1			a	b	x	MC			a	c	MC3
<u>a</u>	<u>b</u>	<u>c</u>				file's			60	70	250
2	1	6	30	40	50	MC1					
250	40	7	5	4	12						
			70	250	7						

3. Write a class to compute the average and variance of a series of numbers. The object will maintain running totals as numbers are added. At the end the average is computed. This is used to compute statistics over time. For example if we want to compute statistics on the size of the groups visiting a museum, we run a program that every time a group arrives the user inserts the group's size into the object. At the end of the day the object is asked to compute the average and variance of the numbers entered. We will use the following formulas:

$$\text{Average} = E(x) = \frac{1}{n} \sum_{i=1}^n x_i \quad \text{Variance} = E(x^2) - E(x)^2 = \frac{1}{n} \sum_{i=1}^n x_i^2 - \left( \frac{1}{n} \sum_{i=1}^n x_i \right)^2$$

So every time we insert we accumulate the sum of  $x$  and  $x^2$  and the number of input seen so far.

Algorithm:

Initialize

```
sum = 0
sum2 = 0
count = 0
```

Insert x:

```
sum = sum + x
sum2 = sum2 + x*x
count++
```

Compute stats

```
average = sum / count
variance = sum2 / count - average*average
```

The constructor is to initialize the variables.

```
class Stats
```

```
{
```

```
private:
```

```
double sum;
```

```
double sum2;
```

```
int count;
```

```
public:
```

```
Stats()
```

```
{
```

```
sum = 0;
```

```
sum2 = 0;
```

```
count = 0;
```

```
}
```

```
void insert(double x)
```

```
{
```

```
sum = sum + x;
```

```
sum2 = sum2 + x * x;
```

```
count++;
```

```
}
```

```
void getStats(double *ave, double *var)
```

```
{
```

```
*ave = sum / count;
```

```
*var = sum2 / count - (*ave) * (*ave);
```

```
}
```

```
}
```