Assist.Prof.Dr. Turgay ALTILAR
Lec.Dr. Yaşar ERENLER
Lec. Binnur KURT

02/06/2006

INTRODUCTION TO SCIENTIFIC AND ENGINEERING COMPUTING
KEYS to FINAL EXAM

(2 Hour-Exam. 3 Pages and 4 Questions. Give your answer inside the corresponding box.)

Q.1) (5) Assume that you have an array defined as

```
int p[]={1,1,2,7,15,…,48} ;
```

How can you read the value of the last element, i.e. the cell storing 48?

p[sizeof(p)/sizeof(int)-1]

Q.2) (25) Which of the following statements are correct? What is the output if wrong statements are discarded?

```
double *p;
double q[10]={0.,1.,2.,3.,4.,5.,6.,7.,8.,9.} ;
```

All statements are CORRECT! NOTHING is WRONG!

C++ version	Output	C version
`cout << *(&q[5]) ;`	**5.0**	`printf("%f",*(&q[5]));`
`cout << *(&q[3]+4) ;`	**7.0**	`printf("%f",*(&q[3]+4));`
`cout << *q ;`	**0.0**	`printf("%f",*q);`
`cout << *(q) ;`	**0.0**	`printf("%f",*(q));`
`cout << *(q+4) ;`	**4.0**	`printf("%f",*(q+4));`
`cout << (&q[8]-&q[3]) ;`	**5.0**	`printf("%d", (&q[8]-&q[3]));`
`cout << *(p=q+3) ;`	**3.0**	`printf("%f",*(p=q+3));`
`cout << (p=q+3)[5] ;`	**8.0**	`printf("%f",(p=q+3)[5]);`

Q.3) (30) Student information is represented by the following structure

```
struct SStudent {
     char name[36]  ;
     long studentID;
     int  grade ;
     char letterGrade[3];
};
typedef struct SStudent  TStudent ;
typedef struct SStudent* PStudent ;
```

a) **(5)** Initialize the array given below with the following values: ("Michael Scofield", 40020336, 97,"AA") – this is 1st student and ("Lincoln Burrows", 40040502, 73, "BB") – this is the 2nd student:

b)

```
TStudent bil105e[]={ {"Michael Scofield", 40020336, 97,"AA"},
                     {"Lincoln Burrows", 40040502, 73, "BB"}
};
```

c) **(25)** Write a function which takes an array of TStudent and its size. The function calculates the letter grades automatically assuming a bell-curve distribution using the following equation where ***m*** denotes the mean and σ denotes the standard deviation. **You are allowed to use ONLY ONE if-statement**. Hint: Use two-dimensional array of size **8×2**.

$$\text{letterGrade}(grade) = \begin{cases} AA & ; m+2.5\sigma \le grade \\ BA & ; m+1.5\sigma \le grade < m+2.5\sigma \\ BB & ; m+0.5\sigma \le grade < m+1.5\sigma \\ CB & ; m-0.5\sigma \le grade < m+0.5\sigma \\ CC & ; m-1.5\sigma \le grade < m-0.5\sigma \\ DC & ; m-2.5\sigma \le grade < m-1.5\sigma \\ DD & ; m-3\sigma \le grade < m-2.5\sigma \\ FF & ; grade < m-3\sigma \end{cases}, \text{ where } \begin{pmatrix} m = \frac{1}{N}\sum_{i=1}^{N} x_i, \\ \sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - m)^2 \end{pmatrix}.$$

```
void computeLetterGrade(PStudent course, int length){
     double mean,sum,sd;
     for (int i=0;i<length;i++)
         sum += course[i].grade;
     mean= sum/length;
     sum=0;
     for (int i=0;i<length;i++)
         sum += pow(course[i].grade-mean,2.);
     sd= sqrt(sum)/length;
     double boundary[7]= {mean+2.5*sd, mean+1.5*sd, mean+0.5*sd,
                          mean-0.5*sd,mean-1.5*sd, mean-2.5*sd,
                          mean-3.0*sd };
     char letter[7][3]={"AA","BA","BB","CB","CC","DC","DD"};
     for (int i=0;i<length;i++){
         strcpy(course[i].letterGrade,"FF");
         for (int j=0;j<7;j++)
             if (course[i].grade>=boundary[j]){
                strcpy(course[i].letterGrade,letter[j]);
                break ;
             }
     }









}
```

Q.4) (40) Write a function which takes a two-dimensional array of type **int** and its size. The function then returns **true** if there is ***NO linearly dependent rows/columns*** and returns **false** ***otherwise***. **Example:** The following matrix has linearly dependent rows and columns:

$$\begin{bmatrix} 2 & 6 & 4 \\ 1 & 3 & 2 \\ 1 & 5 & 2 \end{bmatrix}$$

If you multiply **the second row** by **2** you get **the first row**, or if you multiply the first column by 2 you obtain **the third column**. If you **call** the function with the matrix given above, then the function should **return** ***false***.

```
bool isLinearlyIndependent (int **mat,int row,int col)
{
   double ratio;
   int k;
   for (int i=0;i<row;i++)
      for (int j=i+1;j<row;j++){
         if (mat[i][0]==0) continue;
         ratio= mat[j][0]/mat[i][0];
         for (k=1;k<col;k++)
            if (mat[j][k]!=(ratio*mat[i][k])) break;
         if (k==col) return false;
      }
   for (int i=0;i<col;i++)
      for (int j=i+1;j<col;j++){
         if (mat[0][i]==0) continue;
         ratio= mat[0][j]/mat[0][i];
         for (k=1;k<row;k++)
            if (mat[k][j]!=(ratio*mat[k][i])) break;
         if (k==row) return false;
      }
   return true;
}
```