Coding (or Script) - 5

```
1  // t5.cpp
2
3  #include <iostream.h>
4  #include <stdlib.h>
5  #include <string.h>
6
7  const int max = 30;
8  const int min = 2;
9
10 //*************************************************************
11
12 class  Teacher{
13         char *pszName;
14         int numStud;
15 public:
16         Teacher(char * szName, int numSt = min); //constructor
17         ~Teacher() { delete [] pszName; }                    //destructor
18
19         const int getNumStud(){return numStud;}              //access function
20         void printTeacherInfo();
21 };
22
23
24 //*************************************************************
25 //Constructor
26 inline    Teacher::Teacher(char * szName, int numSt)
27     {
28         pszName=new char [strlen(szName)+1];
29         strcpy(pszName,szName);
30
31         numStud=numSt;
32     }
33
34 //*************************************************************
35
36 inline    void Teacher::printTeacherInfo()
37     {
38        cout << "****************************************\n\n";
39        cout << "Teacher: " << pszName << '\n';
40        cout << "Num of Students:" << numStud << "\n\n";
41        cout << "****************************************\n";
42     }
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
```



```
61 //*************************************************************
62 class Student{
63         char  *name;
64         float quiz1;
65         float quiz2;
66         float final;
67         float lab;
68         float average;
69         float getScore();               // Get score and validate range.
70 public:
71         Student(){ name = 0; }          //NULL pointer so delete will work.
72         ~Student(){ delete [] name; }
73
74         const char * getName(){return name;}     //access function
75         const float getAvg(){return average;}    //access function
76
77         void getStudentInfo() ;
78         void printStudentInfo();
79 };
80
81 //*************************************************************
82 inline        float Student::getScore()      // Get score and validate range.
83         {
84            char temp[10];
85            cin.getline(temp,10);
86
87            while( atof(temp) > 100 || atof(temp) <= 0 ){
88                 cout << " Score entered is out of range." << endl;
89                 cout << " Score must be greater than 0 and less than 100\n";
90                 cout << " Please try again:";
91                 cin.getline(temp,10);
92            }
93            return(atof(temp));
94         }
95
96 //*************************************************************
97 inline        void Student::getStudentInfo()
98         {
99            char szName[30];
100           cout << "Enter Student's name: ";
101           cin.getline(szName,30);
102           name = new char[strlen(szName)+1];
103           strcpy(name, szName);
104
105           cout << "Quiz1: ";
106           quiz1=getScore();
107           cout << "Quiz2: ";
108           quiz2=getScore();
109           cout << "Final: ";
110           final=getScore();
111           cout << "Lab: ";
112           lab=getScore();
113
114           average=(quiz1+quiz2+lab+final) / 4;
115        }
116
117
118
119
120
```

```
121 //*****************************************************************
122 inline        void Student::printStudentInfo()
123         {
124            cout << "Student:    " << name    << '\n';
125            cout << "Quiz1:      " << quiz1   << '\n';
126            cout << "Quiz2:      " << quiz2   << '\n';
127            cout << "Lab:        " << lab     << '\n';
128            cout << "Final:      " << final   << "\n\n";
129            cout << "Average:    " << average << "\n\n";
130         }
131 //*****************************************************************
132 class  ClassStat{
133     Student *pHigh;
134     Student *pLow;
135     float average;
136     float top;
137     float bottom;
138 public:
139     ClassStat() {top = 0; bottom = 100; average = 0;}    //constructor
140
141     void collect_stat(Student *pS);
142
143     void print_class_stat(int numStud);
144 };
145
146 //*****************************************************************
147 inline    void ClassStat::print_class_stat(int numStud)
148     {
149         cout << "*****************************************\n\n";
150         cout << "Highest Score:" << top << "\n";
151         pHigh->printStudentInfo();
152
153         cout << "Lowest Score:" << bottom << "\n";
154         pLow->printStudentInfo();
155
156         cout << "Class average:" << average/ numStud << endl;
157         cout << "*****************************************\n";
158     }
159
160 //*****************************************************************
161 inline    void ClassStat::collect_stat(Student *pS)
162     {
163        float score = pS->getAvg();
164        average=average+score;
165        if (score > top)
166        {
167           pHigh=pS;
168           top=score;
169        }
170        if (score < bottom)
171        {
172           pLow=pS;
173           bottom=score;
174        }
175     }
176
177
178
179
180
```



```
181 //*****************************************************************
182 void main()
183 {
184
185     Student stud[30];
186
187     char tName[30];
188     cout << "Enter teacher's name :";
189     cin.getline(tName,30);
190     cout << endl;
191
192     //create a Teacher object dynamically
193     Teacher * pTeach = new Teacher(tName, 2);
194
195     ClassStat stat;                      //define a ClassStat object
196
197     int numStud=pTeach->getNumStud();    //invoke the Teacher class function.
198
199 //  Student *stud = new Student[numStud];    //better approach than declaring
200 //                                           // an array
201     for (int i=0; i<numStud; i++)
202     {
203        stud[i].getStudentInfo();
204        stat.collect_stat( &stud[i] );
205     }
206
207     cout << "\n\n\n\n\n";
208     pTeach->printTeacherInfo();
209
210     for (i=0; i<numStud; i++)
211     {
212        stud[i].printStudentInfo();
213     }
214
215     stat.print_class_stat(numStud);
216     delete pTeach;
217 //  delete [] stud;   //if stud is allocated dynamically using 'new' operator.
218 }
```