```
//enum1.cpp
#include <iostream>
using namespace std;
enum triangleType{scalene, isosceles, equilateral, noTriangle};
typedef double sideLength;
triangleType triangleShape(double side1, double side2, double side3);
void printShape(triangleType triangle);
int main()
{
        sideLength lenSide1, lenSide2, lenSide3; // use of typedef

        cout<<"Enter the lengths of three sides of a triangle"<<endl;
        cin>>lenSide1>>lenSide2>>lenSide3;
        cout<<endl;

        cout<<"The shape of the triangle is: ";
        printShape(triangleShape(lenSide1,lenSide2,lenSide3));
        cout<<endl;

        return 0;
}
triangleType triangleShape(double side1, double side2, double side3)
{
        if(side1 == side2 && side2 == side3)
                return equilateral;
        else
                if((side1 + side2 >= side3) &&
                   (side1 + side3 >= side2) &&
                   (side2 + side3 >= side1))
                        if(side1 == side2 || side2 == side3 || side1 == side3)
                                return isosceles;
                        else
                                return scalene;
                else
                        return noTriangle;
}

void printShape(triangleType triangle)
{
        switch(triangle)
        {
        case scalene: cout<<"scalene"<<endl;
                        break;
        case isosceles:cout<<"isosceles"<<endl;
                        break;
        case equilateral:cout<<"equilateral"<<endl;
```

```
                        break;
        case noTriangle:cout<<"noTriangle"<<endl;
                        break;
        }
}
```

Output

```
Enter the lengths of three sides of a triangle
12
24
36

The shape of the triangle is: scalene
```

```
// string1.cpp
// illustrates functions for the class string
        #include <string>
        #include <iostream>

using namespace std;

        int main()
        {
                string::size_type positionInString;
                string firstString = "The rain in Spain";
                string secondString;
                cout << "test" << endl;

                cout << firstString.length() << endl;
                cout << firstString.size() << endl;

                positionInString = firstString.find("rain");  // returns position with string
                cout << positionInString << endl ;

                positionInString = firstString.find("falls");  // returns position with string
                if (positionInString == string::npos)
                                cout << "String not found" << endl ;
                else
                                cout << positionInString << endl ;

                secondString = firstString.substr(4,5);  // (starting, number of characters;
                cout << secondString << endl ;

                secondString = firstString.substr(4,66); // (starting, number of characters;
                cout << secondString << endl ;
```

```
              }
 Output
test
17
17
4
String not found
rain
rain in Spain

 //                  string2.cpp
//
// Illustrates a problem with strcpy

// string2.cpp
// illustrates functions for the class string
          #include <string.h>
          #include <iostream>


using namespace std;

          int main()

{
          char myString[]="1234567890";
          char yourString[8];
          strcpy(yourString,myString);
          cout << yourString;
}

no output

// Illustrates a problem with strcpy

// string3.cpp
// illustrates functions for the class string
          #include <string>
          #include <iostream>

using namespace std;

          int main()
           {
          string myString;
```

```
            string yourString="Isn't this fun";
            string anotherString("The rain in Spain");


            myString=yourString;
            cout << myString << endl;
            cout << anotherString << endl;
            anotherString=myString+yourString;
            cout << anotherString<< endl;
            anotherString=myString+ " " + yourString;
            cout << anotherString<< endl;

            cout << "Please enter a string"<< endl;
            getline(cin,myString);  // read until new line
            cout << myString << endl;

            getline(cin,myString,'c'); // read until 'c'
            cout << myString << endl;


            cout << myString[2];
            cout << myString.at(2);
            cout << myString[13];
//          cout << myString.at(13); // will cause the program to error
            return 0;
             }
```

```
output
Isn't this fun
The rain in Spain
Isn't this funIsn't this fun
Isn't this fun Isn't this fun
Please enter a string
abcde
abcde
aabbc
aabb
bb,   // why a ,
```

```
// shows days from start of year to date specified
#include <iostream>
using namespace std;
```

```
int main()
        {
        int month, day, totalDays;
        int daysPerMonth[12] = { 31, 28, 31, 30, 31, 30,
                                 31, 31, 30, 31, 30, 31 };

        cout << "\nEnter month (1 to 12): ";  // get date
        cin >> month;
        cout << "Enter day (1 to 31): ";
        cin >> day;
        totalDays = day;                 // separate days
        for(int index=0; index<month-1; index++)        // add days each month
                totalDays += daysPerMonth[index];
        cout << "Total days from start of year is: " << totalDays;
        }
```

output

```
Enter month (1 to 12): 12
Enter day (1 to 31): 24
Total days from start of year is: 358
```

```
// array2.cpp
// illustrates array operations
#include <iostream>
using namespace std;
#define maximumCells 5
        void printArray(int array[], int numberOfCells) ;
        int main()
        {
        int firstArray[maximumCells];
        int secondArray[maximumCells];
        int index;

//to input array elements

//      cout << firstArray ;  invalid  (no aggregate operations)

        cout << "Please enter 5 numbers" << endl;
        for (index=0;index<maximumCells;index++)
        cin >> firstArray[index];
        printArray(firstArray,maximumCells);


// to copy arrays
//      firstArray=secondArray;   invalid   (no aggregate operations)

        for (index=0;index<maximumCells;index++)
        secondArray[index]=firstArray[index];
        printArray(secondArray,maximumCells);

// to add array elements
//      firstArray=firstArray+secondArray;   invalid  (no aggregate operations)

        for (index=0;index<maximumCells;index++)
        firstArray[index]=firstArray[index]+ secondArray[index];
        printArray(firstArray,maximumCells);
        }

         void printArray(int array[],int numberOfCells)
                {
                  int index;
                  cout << "The current array:" << endl;
                  for (index=0;index < numberOfCells ;index++)
                  cout << array[index] << endl;
                }
```
output

```
Please enter 5 numbers
 2 4 6 8 10
The current array:
2
4
6
8
10
The current array:
2
4
6
8
10
The current array:
4
8
12
16
20
```

```
// array3.cpp
// Aggregate C String I/O in C++
#include <iostream>
using namespace std;
int main()
{
        char   message [ 80 ] ;
        cin  >>  message ;
        cout  <<  message  << endl;  // only valid with strings

        int index=0;
        while (message[index] != '\0')
                cout << message[index++];
}
```

output

```
this is a string
this
this
```

```
// array4.cpp
// array4.cpp
// illustrates const functions and passing arrays
#include <iomanip>
#include <iostream>
using namespace std;


void  Obtain ( int [ ], int ) ;                                    // prototypes here
void  FindWarmest ( const  int[ ],  int , int & ) ;
void  FindAverage  ( const  int[ ],  int , int & ) ;
void  Print ( const  int [ ], int ) ;



int main (   )
{
            int    temp[31] ;       // array to hold up to 31 temperatures
            int    numDays ;
            int    average ;
            int    hottest ;

            cout  <<  "How many daily temperatures? " ;
            cin  >>  numDays ;

            Obtain( temp, numDays ) ;    // call passes value of numDays and
                                                          // address of array temp to function

            cout  <<  numDays  <<  " temperatures"  << endl ;
            Print ( temp, numDays ) ;

            FindAverage ( temp, numDays, average ) ;
            FindWarmest ( temp, numDays, hottest ) ;

            cout  <<  endl  <<  "Average was:  " << average  << endl ;
            cout  <<  "Highest was:  "  << hottest  << endl ;

            return 0 ;
}

void Obtain (  /* out */  int  temp [ ] ,
                                        /* in */  int  number  )

// Has user enter number temperature values at keyboard
// Precondition:
//    number is assigned  &&  number > 0
// Postcondition:
//    temp [ 0 . . number -1 ]  are assigned
{
                    int  m;

                    for ( m = 0 ; m < number;  m++ )
                    {
                    cout << "Enter a temperature : " ;
                                cin >>  temp [m] ;
                     }
}
void Print (  /* in */  const  int  temp [ ] ,
                                        /* in */  int  number  )

// Prints number  temperature values to screen
// Precondition:
//    number is assigned  &&  number > 0
//    temp [0 . . number -1 ] are assigned
// Postcondition:
//    temp [ 0 . . number -1 ]  have been printed 5 to a line
{
                    int  m;
                    cout  <<   "You entered: " ;
                    for ( m = 0 ; m < number;  m++ )
                      {
                                if  ( m % 5 == 0 )
                                                 cout  <<  endl ;
                    cout  <<  setw(7) << temp [m] ;
                      }
}
void  FindAverage (  /* in */   const  int  temp [ ] ,
                                                                                                                        /* in */   int
number ,
                                                                                                                       /* out */  int
&  avg )

// Determines average of temp[0 . . number-1]
// Precondition:
//       number is assigned  &&  number > 0
//       temp [0 . . number -1 ] are assigned
// Postcondition:
//       avg == arithmetic average of temp[0 . . number-1]
{
                    int  m;
```

```
                int  total = 0;
                for ( m = 0 ; m < number;  m++ )
                 {
                        total = total + temp [m] ;
                 }
                avg = int (float (total) / float (number) + .5) ;
}
void  FindWarmest (  /* in */    const int   temp [ ] ,
                                                                                              /* in
*/   int   number ,
                                                                                        /* out */
int &   largest )

// Determines largest of temp[0 . . number-1]
// Precondition:
//        number is assigned  &&  number > 0
//        temp [0 . . number -1 ] are assigned
// Postcondition:
//        largest== largest value in temp[0 . . number-1]
{
                int  m;
        //      temp[0]=11;  will cause a compliation error
                largest = temp[0] ;      // initialize largest to first element
                                                      // then compare with other elements
                for ( m = 0 ; m < number;  m++ )
                 {
                        if ( temp [m]  > largest )
                                    largest  =  temp[m] ;
                 }
}
```

```
output
How many daily temperatures? 2
Enter a temperature : 20
Enter a temperature : 90
2 temperatures
You entered:
         20     90
Average was:  55
Highest was:  90
```

cstring1.cpp

```
// Aggregate C String I/O in C++

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

```
int main()
{
        char   message [ 80 ] ;
      cin  >>  message ;
        cout  <<  message  << endl;  // only valid with strings

        int index=0;
        while (message[index] != '\0')
        cout << message[index++];
        return 0;

}
```

output

```
This is a test
This
This
```

```
// cstring2.cpp
/ Aggregate C String I/O in C++

#include <iostream>
using namespace std;
int main()
{

        char  fullName [ 32 ] ;
        char  address [ 32 ] ;
        char  school [100] ;
        char singleChar;

        cout << "Please enter a single character: " ;
        cin.get(singleChar);
        cout << singleChar << endl;
        cin.get(singleChar); // consume newline character

        cout << "Enter your full name: " ;
        cin.get ( fullName, 31 ) ;
        cout << fullName[0] << endl ; // can access a character at a time
        cout << fullName << endl ;
        cin.get(singleChar); // consume newline character

        cout << "Enter your address: " ;
        cin.get ( address, 31 ) ;
        cin.get(singleChar); // consume newline character

        cout << "Please enter your school name: " ;
```

```
          cin.ignore (5,' ') ;  // ignore the first five characters
          cin.get ( school, 40 ) ;  // read another line of data
          cout << school << endl ;

}
```

output

```
Please enter a single character: X
X
Enter your full name: Jones
J
Jones
Enter your address: 1234 main street
Please enter your school name: Saddleback
eback
```

```
// cstring3.cpp
// reading and writing to files

#include <iostream>
#include <fstream>
using namespace std;


int main()
{

          int firstNumber;
          int secondNumber;
          int thirdNumber;


          ifstream inFile;   // file stream for input file
          ofstream outFile;  // file stream for output file

          //open the input and output file
          inFile.open("input");
          if ( !inFile )
                    {
                              cout << "can't open input file" << endl;
                              return 1;
                    }

          cout << "open of input file successful" << endl;

          outFile.open("myOutput.doc");
          if ( !outFile )
                    {
                              cout << "can't open output file" << endl;
                              return 1;
                    }

          cout << "open of output file successful" << endl;

          inFile >> firstNumber >> secondNumber >> thirdNumber ;
          cout << firstNumber << " " << secondNumber << " " << thirdNumber << endl ;
          outFile << firstNumber << secondNumber <<  thirdNumber << endl;

           return 0;
 }
```

output

```
open of input file successful
open of output file successful
10 20 30
```

```
//                            cstring4.cpp
//  Objective -  C string functions

#include <iostream>
#include <cstring>

using namespace std;

#define MAXSTRING 10
int main()
{
          char helloString[MAXSTRING] = "Hello";
          char helloString2[MAXSTRING] = "Hello";
          char byeString[MAXSTRING] = "Bye";

          cout << "sizeof " << sizeof("I am here") << endl;
          cout << "strlen " << strlen("I am here") << endl;

          if (helloString==helloString2)
                    cout << "helloString==helloString2" << endl ;

          if (strcmp(helloString,helloString2)==0)
                     cout << "The contents of helloString and helloString2 are the same"
                     << endl ;

          strcpy(helloString2,byeString);
```

```
        cout <<  "helloString2 is " <<  helloString2 << endl ;

        strcat(helloString,byeString);
        cout <<  "helloString is " <<  helloString << endl ;

        return 0;
}
```
output
```
sizeof 10
strlen 9
The contents of helloString and helloString2 are the same
helloString2 is Bye
helloString is HelloBye
```

```
//                    cstring5.cpp
//  atoi and atof functions
// data must match

#include <iostream>
#include <cstdlib>
using namespace std;

#define MAXSTRING 10
int main()
{
        char inputString[MAXSTRING];

                cin >> inputString;
                cout << atoi(inputString) << endl;

                cin >> inputString;
                cout << atoi(inputString) << endl;

                cin >> inputString;
                cout << atof(inputString) << endl;

                cin >> inputString;
                cout << atof(inputString) << endl;

        return 0;
}
```
output
```
123456
123456
abc
0
```

```
1.234
1.234
invalid
0
```

```
// marray1.cpp
// displays sales chart, initializes 2-d array
#include <iostream>
using namespace std;

const int DISTRICTS = 4;       // array dimensions
const int MONTHS = 3;

int main()
   {
          int district, month;
                                                    // initialize array elements
          float sales[DISTRICTS][MONTHS]
                      = {  {  1432.07,   234.50,   654.01 },
                                        {   322.00, 13838.32, 17589.88 },
                                        {  9328.34,   934.00,  4492.30 },
                                        { 12838.29,  2332.63,    32.93 }  };

          for(district=0; district<DISTRICTS; district++)
                    {
                    cout <<"\nDistrict " << district+1  << endl;
                    for(month=0; month<MONTHS; month++)
           cout << sales[district][month]  << endl;  // access array element
                    }
          return 0;
          }
```

output

```
District 1
1432.07
234.5
654.01

District 2
322
13838.3
17589.9

District 3
9328.34
934
4492.3
```

```
District 4
12838.3
2332.63
32.93
```

```
// marray2.cpp
#include <iostream>
using namespace std;


void printArray( int rows, int columns, int array [][5] );

int main()
{
	int twoDimArray[4][5],
	     row, column;

	for ( row = 0; row < 4; row++ )
		for ( column = 0; column < 5; column++ )
			twoDimArray[row][column] = row * column;

	printArray( 4, 5, twoDimArray );
}

//*************************************   printArray()   ***********/
//   An output routine. Displays the contents of an array of
//   type int. The array is passed as a parameter along with
//   the number of rows and columns to be displayed.

void printArray( int rows, int columns, int array[][5] )
{
	int i = 0, j = 0;

	while ( i < rows )
	 {
		cout << array[i][j];

		( j == columns-1 ) ? cout << '\n' : cout << '\t' ;

		( j == columns - 1 ) ? i++,j=0 : j++;
	}
}
```

output

```
0	0	0	0	0
```

0	1	2	3	4
0	2	4	6	8
0	3	6	9	12

```
// marray3.cpp
// illustrates passing two dimensional arrays
//
//   Objective - Demonstrates multidimensional arrays and
//                     the?: construct.

//
// Illustrates using three-dimensional arrays

#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
        double triple[2][3][4] = {
                                    { { 0.0, 0.1, 0.2, 0.3 },
                                            { 1.0, 1.1, 1.2, 1.3 },
                                            { 2.0, 2.1, 2.2, 2.3 }
                                    },
                                    { { 10.0, 10.1, 10.2, 10.3 },
                                            { 11.0, 11.1, 11.2, 11.3 },
                                            { 12.0, 12.1, 12.2, 12.3 }
                                    }
                                };
        int index;
        int jndex;
        int kndex;

        cout << fixed << showpoint << setprecision(1) ;

        for ( index = 0; index < 2; index++ )
         {
                for (  jndex = 0;  jndex < 3; jndex++ )
                 {
                        for ( kndex = 0; kndex < 4; kndex++ )
                         {
                                cout << setw(5) << triple[index][jndex][kndex];
                         }
                        cout << endl;
                 }
                cout << endl << endl;
          }
}
```

output
```
  0.0  0.1  0.2  0.3
  1.0  1.1  1.2  1.3
  2.0  2.1  2.2  2.3


 10.0 10.1 10.2 10.3
 11.0 11.1 11.2 11.3
 12.0 12.1 12.2 12.3
```

```
// bsearch.cpp
#include <iostream>

using namespace std;

int binarySearch(const int list[], int listLength, int searchItem);

int main()
{
	int list[] = {2,5,10,16,25,34,46,56,73,89};
	int location;

	location = binarySearch(list,10,56);

	if(location != -1)
		cout<<"Item found at location "<<location<<endl;
	else
		cout<<"Item not in the list"<<endl;

	return 0;
}

int binarySearch(const int list[], int listLength, int searchItem)

{
	int first = 0;
	int last = listLength - 1;
	int mid;

	bool found = false;

	while(first <= last && !found)
	{
		mid = (first + last) / 2;
```

```
                    if(list[mid] == searchItem)
                              found = true;
                    else
                        if(list[mid] > searchItem)
                                last = mid - 1;
                        else
                                first = mid + 1;
          }

          if(found)
             return mid;
          else
             return -1;
}//end binarySearch
```

output

```
Item found at location 7
```

```
//cast1
// This program illustrates how explicit type conversion works.

#include <iostream>

using namespace std;

int main()
{
    cout << "static_cast<int>(7.9) = " << static_cast<int>(7.9)
         << endl;
    cout << "static_cast<int>(3.3) = " << static_cast<int>(3.3)
         << endl;
    cout << "static_cast<double>(25) = " << static_cast<double>(25)
         << endl;
    cout << "static_cast<double>(5 + 3) = "
         << static_cast<double>(5 + 3)
         << endl;
    cout << "static_cast<double>(15) / 2 = "
         << static_cast<double>(15) / 2
         << endl;
    cout << "static_cast<double>(15 / 2) = "
         << static_cast<double>(15 / 2)
         << endl;
    cout << "static_cast<int>(7.8 + static_cast<double>(15) / 2) = "
         << static_cast<int>(7.8 + static_cast<double>(15) / 2)
         << endl;
    cout << "static_cast<int>(7.8 + static_cast<double>(15 / 2)) = "
         << static_cast<int>(7.8 + static_cast<double>(15 / 2))
```

```
        << endl;

    return 0;
}
```

Output

```
static_cast<int>(7.9) = 7
static_cast<int>(3.3) = 3
static_cast<double>(25) = 25
static_cast<double>(5 + 3) = 8
static_cast<double>(15) / 2 = 7.5
static_cast<double>(15 / 2) = 7
static_cast<int>(7.8 + static_cast<double>(15) / 2) = 15
static_cast<int>(7.8 + static_cast<double>(15 / 2)) = 14
```

****

```
// struct1.cpp
// uses parts inventory to demonstrate structures
#include <iostream>
using namespace std;

struct part          // specify a structure
   {
   int modelNumber;   // ID Number of widget
   int partNumber;    // ID Number of widget part
   float cost;        // cost of part
   };

int main()
   {
   part part1;               // define a structure variable

   part1.modelNumber = 6244;  // give values to structure members
   part1.partNumber = 373;
   part1.cost = 217.55;
                                 // display structure members
   cout << "\nModel "  << part1.modelNumber;
   cout << ", part "   << part1.partNumber;
   cout << ", costs $" << part1.cost;
   }
```

output

```
Model 6244, part 373, costs $217.55
```

```
// struct2.cpp
// to illustrate structs within structs
//
```

```
        #include <iostream>
using namespace std;
        struct dateType
        {
                int month;
                int day;
                int year;
        } ;

        struct statisticsType
        {
                float failRate;
                dateType lastServiced;
                int downDays;
        }  ;

        struct machineRec
        {
                int idNumber;
           string description;
                statisticsType history;
                dateType purchaseDate;
                float cost;
        } ;

        int main()
        {
          machineRec myMachine;
          myMachine.idNumber = 123 ;
          myMachine.description = "my description" ;
          myMachine.history.failRate = 10 ;
          myMachine.history.lastServiced.month = 1 ;
          myMachine.history.lastServiced.day = 2 ;
          myMachine.history.lastServiced.year = 2014 ;
          myMachine.purchaseDate.month = 2 ;
          myMachine.purchaseDate.day = 3 ;
          myMachine.purchaseDate.year = 2005 ;
          myMachine.cost = 123 ;

          cout << myMachine.idNumber << endl;
          cout << myMachine.description << endl;
          cout << myMachine.history.failRate << endl;
          cout << myMachine.history.lastServiced.month << endl;
          cout << myMachine.history.lastServiced.day << endl ;
          cout << myMachine.history.lastServiced.year << endl;
          cout << myMachine.purchaseDate.month << endl ;
```

```
            cout << myMachine.purchaseDate.day << endl ;
            cout << myMachine.purchaseDate.year << endl ;
            cout << myMachine.cost << endl;
  }
```

output

```
123
my description
10
1
2
2014
2
3
2005
123
```

```
/*                              struct3.cpp
 *
 *   Synopsis  -  Accepts input of an autoPart from standard
 *                      input and echoes it to standard output.
 *
 *   Objective -  To illustrate passing a structure using references and
 *                   by copy
 */

/* Include Files */
#include <iostream>
#include <string.h>
#include <cstdlib>
#include <cstdio>
using namespace std;


/* Constant Declarations */
#define GOT_ONE         1
#define NONE_ENTERED    0
#define IDSIZE                  8

/* Type descriptions */
struct autoPart {
          char id[8];
          float price;
          int currentInventory;
};
```

```
/* Function Declarations */
void putPart(autoPart part);
int getPart(autoPart& Part);

int main()
{
        cout << "auto parts" << endl;
        autoPart part;
        int retval;

        retval = getPart( part );
        if ( retval == GOT_ONE )
                putPart( part );
}

/************************************   getPart()    *******/
/*   Accepts input of a struct autoPart from standard input
 *   and returns input values in its parameter.
 */

int getPart(autoPart& part )
{
        char instring[512];

        cout << "Enter the part number : ";
        cin >> instring;
        if ( strlen( instring ) > 0 ) {
                strncpy( part.id, instring, 7);
                part.id[7] = '\0';

                cout << "Enter the price: ";
                cin >> instring;
                part.price = atof( instring  );

                cout << "Enter the amount in inventory : " ;
                cin >> instring;
                part.currentInventory = atoi(instring);
                return( GOT_ONE );
        }
        else
                return( NONE_ENTERED );
}

/***********************************    putPart()    ********/
/*   outputs contents of a struct autoPart to the terminal    */
void putPart(autoPart part )
```

```
{
        cout << "Part-id: " <<  part.id << endl;
        cout << "Price : " << part.price << endl;
        cout << "Quantity: " << part.currentInventory << endl;
}
```

output

```
auto parts
Enter the part number : 12345
Enter the price: 25
Enter the amount in inventory : 45
Part-id: 12345
Price : 25
Quantity: 45
```

```
// struct4.cpp
// structure variables as array elements
#include <iostream>
using namespace std;


const int SIZE = 2;            // number of parts in array

struct part                    // specify a structure
  {
        int modelnumber;             // model number of widget
        int partnumber;              // part number of widget part
        int quantityPerBox [2];       // quantity in a box (2 types)
        float cost;                  // cost of part
  };

int main()
  {
        int index;
  part apart[SIZE];            // define array of structures

        for(index=0; index<SIZE; index++)        // get values for all members
    {
    cout << endl;
    cout << "Enter model number: ";
                cin >> apart[index].modelnumber;      // get model number
                cout << "Enter part number: ";
                cin >> apart[index].partnumber;       // get part number
                cout << "Enter quantity for type 1 box: ";
                cin >> apart[index].quantityPerBox[0]; // get quantity per type 1 box
                cout << "Enter quantity for type 2 box: ";
                cin >> apart[index].quantityPerBox[1]; // get quantity per type 2 box
```

```
                    cout << "Enter cost: ";
                    cin >> apart[index].cost;           // get cost
       }
           for(index=0; index<SIZE; index++)        // show values for all members
       {
                    cout << "\nModel " << apart[index].modelnumber;
                    cout << "  Part "  << apart[index].partnumber;
                    cout << "  Type 1 Quantity "  << apart[index].quantityPerBox[0];
                    cout << "  Type 2 Quantity "  << apart[index].quantityPerBox[1];
                    cout << "  Cost "  << apart[index].cost;
       }
    }
  output
```

```
Enter model number: 123545
Enter part number: 123
Enter quantity for type 1 box: 2
Enter quantity for type 2 box: 6
Enter cost: 35.88

Enter model number: 12
Enter part number: 123
Enter quantity for type 1 box: 5
Enter quantity for type 2 box: 7
Enter cost: 55.77

Model 123545  Part 123  Type 1 Quantity 2  Type 2 Quantity 6  Cost 35.88
Model 12  Part 123  Type 1 Quantity 5  Type 2 Quantity 7  Cost 55.77
```

```
// struct5.cpp
// uses parts inventory to demonstrate structures
#include <iostream>
using namespace std;

struct part          // specify a structure
  {
  int modelNumber;   // ID Number of widget
  int partNumber;    // ID Number of widget part
  float cost;        // cost of part
  void print ()
    {
        cout << "print cost "<< cost << endl;
    }
  };

int main()
```

```
  {
  part part1;              // define a structure variable

  part1.modelNumber = 6244;  // give values to structure members
  part1.partNumber = 373;
  part1.cost = 217.55;
                               // display structure members
  cout << "\nModel "  << part1.modelNumber;
  cout << ", part "   << part1.partNumber;
  cout << ", costs $" << part1.cost << endl;
  part1.print();
  }
 output
```

```
Model 6244, part 373, costs $217.55
print cost 217.55
```

```
//class1.h, the specification  file for the class clockType
#ifndef CLASS1_H_
#define CLASS1_H_

//class1.h, the specification  file for the class clockType
class clockType
{
public:
    void setTime(int hours, int minutes, int seconds);
                    //Function to set the time
                    //Post: time is set according to the
                    //parameters: hr = hours; min = minutes;
                    //                sec = seconds

          void getTime(int& hours, int& minutes, int& seconds);
                    //Function to return the time
                    //Post: hours = hr; minutes = min;
                    //                                    seconds = sec;

    void printTime() const;
                    //Function to print the time
                    //Time is printed in the form hh:mm:ss

    void incrementSeconds();
                    //Function to increment the time by 1 second
                    //Post: The time is incremented by 1 second
                    //If the before-increment time is 23:59:59, the time
                    //is reset to 00:00:00
```

```
   void incrementMinutes();
                  //Function to increment the time by 1 minute
                  //Post: The time is incremented by 1 minute
                  //If the before-increment time is 23:59:53, the time
                  //is reset to 00:00:53

   void incrementHours();
                  //Function to increment the time by 1 hour.
                  //Post: The time is incremented by 1 hour.
                  //If the before-increment time is 23:45:53, time
                  //is reset to 00:45:53

   bool equalTime(const clockType& otherClock) const;
                  //Function to compare the two times
                  //Function returns true if this time is equal to
                  //otherClock; otherwise it returns false

   private:
   int hr;  //store hours
   int min; //store minutes
   int sec; //store seconds
};
#endif /* CLASS1_H_ */
//class1.cpp
#include <iostream>

#include "class1.h"

using namespace std;

void clockType::setTime(int hours, int minutes, int seconds)
{
        if(0 <= hours && hours < 24)
          hr = hours;
        else
          hr = 0;

        if(0 <= minutes && minutes < 60)
          min = minutes;
        else
          min = 0;

        if(0 <= seconds && seconds < 60)
          sec = seconds;
        else
          sec = 0;
```

```
}

void clockType::getTime(int& hours, int& minutes, int& seconds)
{
        hours = hr;
        minutes = min;
        seconds = sec;
}

void clockType::incrementHours()
{
        hr++;
        if(hr > 23)
          hr = 0;
}

void clockType::incrementMinutes()
{
        min++;
        if(min > 59)
        {
          min = 0;
          incrementHours();
        }
}

void clockType::incrementSeconds()
{
        sec++;

        if(sec > 59)
        {
          sec = 0;
          incrementMinutes();
        }
}

void clockType::printTime() const
{
        if(hr < 10)
          cout<<"0";
        cout<<hr<<":";

        if(min < 10)
          cout<<"0";
        cout<<min<<":";
        if(sec < 10)
          cout<<"0";
        cout<<sec;
}

bool clockType::equalTime(const clockType& otherClock) const
{
        return (hr == otherClock.hr
                   && min == otherClock.min
                   && sec == otherClock.sec);
}

//class1.cpp
#include <iostream>

#include "class1.h"

using namespace std;

int main()
{
        clockType myClock;
        clockType yourClock;

        int hours;
        int minutes;
        int seconds;

        myClock.setTime(5,4,30);                                   //Line 1
   cout<<"Line 2: myClock: ";                                      //Line 2
        myClock.printTime();                                                //Line 3
        cout<<endl;                                                                  //Line
4

        cout<<"Line 5: yourClock: ";                               //Line 5
        yourClock.printTime();                                              //Line 6
        cout<<endl;                                                                  //Line
7

        yourClock.setTime(5,45,16);                                         //Line 8

        cout<<"Line 9: After setting - yourClock: ";       //Line 9
        yourClock.printTime();                                              //Line 10
        cout<<endl;                                                                  //Line
11
```

```
        if(myClock.equalTime(yourClock))                      //Line 12
          cout<<"Line 13: Both times are equal."
                  <<endl;                                          //Line 13
        else                                                       //Line 14
          cout<<"Line 15: The two times are not equal"
                  <<endl;                                          //Line 15

        cout<<"Line 16: Enter hours, minutes, and "
           <<"seconds: ";                                          //Line 16
        cin>>hours>>minutes>>seconds;                         //Line 17
        cout<<endl;                                                //Line 18

        myClock.setTime(hours,minutes,seconds);               //Line 19

        cout<<"Line 20: New myClock: ";                       //Line 20
        myClock.printTime();                                       //Line 21
        cout<<endl;                                                //Line 22

        myClock.incrementSeconds();                                //Line 23

        cout<<"Line 24: After incrementing the clock by "
                <<"one second, myClock: ";                         //Line 24
        myClock.printTime();                                       //Line 25
   cout<<endl;                                                     //Line 26

        return 0;
}//end main
```

output

```
Line 2: myClock: 05:04:30
Line 5: yourClock: 0-2:1965035874:1965382596
Line 9: After setting - yourClock: 05:45:16
Line 15: The two times are not equal
Line 16: Enter hours, minutes, and seconds: 12
24
49

Line 20: New myClock: 12:24:49
Line 24: After incrementing the clock by one second, myClock: 12:24:50
```

```
//class2.cpp
/**********************************************************
// header and implementation files same as class1 example
//*********************************************************
#include <iostream>
```

```
#include "class1.h"

using namespace std;

int main()
{
	clockType myClock;
	clockType yourClock;

	int hours;
	int minutes;
	int seconds;

	//*****************************************************
	// hr=5 // not defined
	//myClock.hr=5 // cannot access private data members

	//if (myClock==yourClock) // illegal aggregate operation
	//	cout << "equal" << endl;
	//*****************************************************

	myClock.setTime(5,4,30);                                  //Line 1
	 cout<<"Line 2: myClock: ";                               //Line 2
	myClock.printTime();                                              //Line 3
	cout<<endl;                                                       //Line 4

	cout<<"Line 5: yourClock: ";                              //Line 5
	yourClock.printTime();                                            //Line 6
	cout<<endl;                                                       //Line 7

	yourClock.setTime(5,45,16);                                       //Line 8

	cout<<"Line 9: After setting - yourClock: ";        //Line 9
	yourClock.printTime();                              //Line 10
	cout<<endl;                                                 //Line 11

	if(myClock.equalTime(yourClock))                          //Line 12
	   cout<<"Line 13: Both times are equal."
	           <<endl;                                                //Line 13
	else                                                              //Line 14
	   cout<<"Line 15: The two times are not equal"
	           <<endl;                                                //Line 15
	cout<<"Line 16: Enter hours, minutes, and "
	    <<"seconds: ";                                                //Line 16
	cin>>hours>>minutes>>seconds;                             //Line 17
	cout<<endl;                                                       //Line 18
```

```
        myClock.setTime(hours,minutes,seconds);                     //Line 19

        cout<<"Line 20: New myClock: ";                             //Line 20
        myClock.printTime();                                            //Line 21
        cout<<endl;                                                     //Line 22

        myClock.incrementSeconds();                                     //Line 23

        cout<<"Line 24: After incrementing the clock by "
                <<"one second, myClock: ";                              //Line 24
        myClock.printTime();                                    //Line 25
         cout<<endl;                                                    //Line 26

        return 0;
}//
```

output

```
Line 2: myClock: 05:04:30
Line 5: yourClock: 0-2:1965035874:1965382596
Line 9: After setting - yourClock: 05:45:16
Line 15: The two times are not equal
Line 16: Enter hours, minutes, and seconds: 10 12 14

Line 20: New myClock: 10:12:14
Line 24: After incrementing the clock by one second, myClock: 10:12:15
```

```
// class3.h
class  MyTime       // declares a  class data type
{                                          //  does not allocate memory

public :                        //  5 public function members

        void        Set ( int  hours , int  mins , int  secs ) ;
        void         Increment ( ) ;
        void         Write ( )  const ;
        bool        Equal ( MyTime   otherTime )  const ;
        bool        LessThan ( MyTime   otherTime )  const ;

private :                             //  3 private data members

        int         hrs ;
        int         mins ;
        int            secs ;
} ;
// class3i.cpp
// class3i.cpp
//   IMPLEMENTATION FILE
//   Implements the MyTime member functions

#include <iostream>
using namespace std;
#include "class3.h"  // also must appear in client code

// private data members
//                  int hrs ;
//                  int mins ;
//                  int secs ;

void  MyTime::Set(int hours, int minutes, int seconds)
         {
                  hrs = hours ;
                  mins = minutes ;
                  secs = seconds ;
         }

void  MyTime::Increment()
         {
                  secs++ ;
                  if (secs > 59)
                           {
                                    secs = 0;
                                    mins++;
                                    if (mins > 59)
                                             {
                                                      mins = 0;
                                                      hrs++;
                                                      if (hrs > 23)
                                                               hrs=0;
                                             }
                           }
         }

void   MyTime :: Write ( )   const

           //   Postcondition:   Time has been output in form HH:MM:SS

         {        if  ( hrs < 10 )
                                    cout << '0' ;
                  cout  << hrs  <<  ':' ;
                  if  ( mins < 10 )
                                    cout << '0' ;
```

```
                    cout  << mins  <<  ':' ;
                    if  ( secs < 10 )
                                        cout << '0' ;
                    cout  << secs ;
          }

bool  MyTime :: Equal ( /* in */  MyTime otherTime ) const
  //  Postcondition:
  //      Function value == true,    if this time equals otherTime
  //                                  == false , otherwise
  {
          return ( (hrs == otherTime.hrs)  && (mins == otherTime.mins) && (secs  ==
otherTime.secs) ) ;
          }

bool  MyTime :: LessThan ( /* in */  MyTime otherTime ) const

  {
                    return ((hrs < otherTime.hrs)   ||
                    (hrs == otherTime.hrs && mins < otherTime.mins) ||
                    (hrs == otherTime.hrs && mins == otherTime.mins
                     &&secs < otherTime.secs));
  }


// class3.cpp
// to illustrate creating and using classes
//
#include <iostream>
using namespace std;

#include "class3.h"

int main()
{
          MyTime startTime;   // create an instance of a class
          startTime.Set(10,20,30) ;
          startTime.Write();
          cout << endl;

          MyTime endTime;   // create an instance of a class
          endTime.Set(12,24,49) ;
          endTime.Write();
          cout << endl;

          endTime.Increment();
```

```
        endTime.Write();
        cout << endl;

        if (startTime.Equal(endTime))
                cout << "times are equal" << endl;
        else
                cout << "times are not equal" << endl;

        // hrs=5;  compilation error
}
```

output

```
10:20:30
12:24:49
12:24:50
times are not equal
```

// class4.h

```
class  MyTime       // declares a  class data type
{                                               //  does not allocate memory

public :                        //  5 public function members

        MyTime ( int  initHrs ,  int  initMins ,  int  initSecs ) ; // constructor
        MyTime ( ) ;                            // default constructor
        void         Set ( int  hours , int  mins , int  secs ) ;
        void          Increment ( ) ;
        void          Write ( )  const ;
        bool         Equal ( MyTime   otherTime )  const ;
        bool         LessThan ( MyTime   otherTime )  const ;

private :                                //  3 private data members

        int          hrs ;
        int          mins ;
        int             secs ;
} ;
```

// class4i.cpp

```
// class4i.cpp

//   IMPLEMENTATION FILE
//   Implements the MyTime member functions

#include <iostream>
using namespace std;
#include "class4.h"  // also must appear in client code
```

```
// private data members
//                int hrs ;
//                int mins ;
//                int secs ;

MyTime :: MyTime (  )
//  Default  Constructor
//  Postcondition:
//                        hrs == 0   &&   mins == 0   &&  secs == 0
{
                hrs  =  0 ;
     mins = 0 ;
                secs = 0 ;
}

MyTime :: MyTime ( /* in */   int   initHrs,
                                /* in */   int   initMins,
                                /* in */   int   initSecs )
//  Constructor
//  Precondition:  0 <= initHrs <= 23    &&    0 <= initMins <= 59
//                         0 <= initSecs <= 59
//  Postcondition:
//                hrs == initHrs  &&  mins == initMins  && secs == initSecs
{
                 hrs  =  initHrs ;
                 mins =  initMins ;
      secs =  initSecs ;
}

void  MyTime::Set(int hours, int minutes, int seconds)
        {
                hrs = hours ;
                mins = minutes ;
                secs = seconds ;
        }

void  MyTime::Increment()
        {
                secs++ ;
                if (secs > 59)
                        {
                                secs = 0;
                                mins++;
                                if (mins > 59)
                                        {
```

```
                                    mins = 0;
                                    hrs++;
                                    if (hrs > 23)
                                            hrs=0;
                                }
                        }
        }

void   MyTime :: Write ( )   const

        //   Postcondition:   Time has been output in form HH:MM:SS

        {       if  ( hrs < 10 )
                                cout << '0' ;
                cout  << hrs  <<  ':' ;
                if  ( mins < 10 )
                                cout << '0' ;
                cout  << mins  <<  ':' ;
                if  ( secs < 10 )
                                cout << '0' ;
                cout  << secs ;
        }

bool  MyTime :: Equal ( /* in */  MyTime otherTime ) const
  //  Postcondition:
  //      Function value == true,    if this time equals otherTime
  //                                 == false , otherwise
  {
                return ( (hrs == otherTime.hrs)  && (mins == otherTime.mins)
                                                      && (secs  == otherTime.secs) ) ;
        }


bool  MyTime :: LessThan ( /* in */  MyTime otherTime ) const

  {
                return ((hrs < otherTime.hrs)   ||
                (hrs == otherTime.hrs && mins < otherTime.mins) ||
                (hrs == otherTime.hrs && mins == otherTime.mins
                 &&secs < otherTime.secs));
  }



// class4.cpp
// class4.cpp
// to illustrate creating and using classes
// with two constructors
        #include <iostream>
using namespace std;
        #include "class4.h"

int main()
{
        MyTime startTime;   // create an instance of a class (default constructor)
  startTime.Write();
        cout << endl;
        startTime.Set(10,20,30) ;
        startTime.Write();
        cout << endl;

        MyTime endTime(12,24,49);   // create an instance of a class
        endTime.Write();
        cout << endl;
        endTime.Set(12,20,59) ;
        endTime.Write();
        cout << endl;

        endTime.Increment();
        endTime.Write();
        cout << endl;

        if (startTime.Equal(endTime))
                cout << "times are equal" << endl;
        else
                cout << "times are not equal" << endl;

        // hrs=5;  compilation error
        //startTime.hrs=5 compliation error
}
```

output

```
00:00:00
10:20:30
12:24:49
12:20:59
12:21:00
times are not equal
```

```
//class5.h, the specification  file for the class clockType
class clockType
{
public:
```

```
void setTime(int hours, int minutes, int seconds);
            //Function to set the time
            //Post: time is set according to the
            //parameters: hr = hours; min = minutes;
            //              sec = seconds

    void getTime(int& hours, int& minutes, int& seconds);
            //Function to return the time
            //Post: hours = hr; minutes = min;
            //                              seconds = sec;

void printTime() const;
            //Function to print the time
            //Time is printed in the form hh:mm:ss

void incrementSeconds();
            //Function to increment the time by 1 second
            //Post: The time is incremented by 1 second
            //If the before-increment time is 23:59:59, the time
            //is reset to 00:00:00

void incrementMinutes();
            //Function to increment the time by 1 minute
            //Post: The time is incremented by 1 minute
            //If the before-increment time is 23:59:53, the time
            //is reset to 00:00:53

void incrementHours();
            //Function to increment the time by 1 hour.
            //Post: The time is incremented by 1 hour.
            //If the before-increment time is 23:45:53, time
            //is reset to 00:45:53

bool equalTime(const clockType& otherClock) const;
            //Function to compare the two times
            //Function returns true if this time is equal to
            //otherClock; otherwise it returns false

clockType(int hours, int minutes, int seconds);
            //Constructor with parameters
            //Post: The time is set according to
            //the parameters
            //  hr = hours; min = minutes; sec = seconds

clockType();
            //Default constructor with parameters
```

```
                 //Post: time is set to 00:00:00
                 //  hr = 0; min = 0; sec = 0

private:
   int hr;  //store hours
   int min; //store minutes
   int sec; //store seconds
};
//class5i.cpp
//class5i.cpp

#include <iostream>
#include "class5.h"

using namespace std;

void clockType::setTime(int hours, int minutes, int seconds)
{
        if(0 <= hours && hours < 24)
          hr = hours;
        else
          hr = 0;

        if(0 <= minutes && minutes < 60)
          min = minutes;
        else
          min = 0;

        if(0 <= seconds && seconds < 60)
          sec = seconds;
        else
          sec = 0;
}

void clockType::getTime(int& hours, int& minutes, int& seconds)
{
        hours = hr;
        minutes = min;
        seconds = sec;
}

void clockType::incrementHours()
{
        hr++;
        if(hr > 23)
          hr = 0;
```

```
}

void clockType::incrementMinutes()
{
        min++;
        if(min > 59)
        {
          min = 0;
          incrementHours();
        }
}

void clockType::incrementSeconds()
{
        sec++;

        if(sec > 59)
        {
          sec = 0;
          incrementMinutes();
        }
}

void clockType::printTime() const
{
        if(hr < 10)
          cout<<"0";
        cout<<hr<<":";

        if(min < 10)
          cout<<"0";
        cout<<min<<":";

        if(sec < 10)
          cout<<"0";
        cout<<sec;
}

bool clockType::equalTime(const clockType& otherClock) const
{
        return (hr == otherClock.hr
                  && min == otherClock.min
                  && sec == otherClock.sec);
}

clockType::clockType(int hours, int minutes, int seconds)
```

```
{
        if(0 <= hours && hours < 24)
                hr = hours;
        else
                hr = 0;

        if(0 <= minutes && minutes < 60)
                min = minutes;
        else
                min = 0;

        if(0 <= seconds && seconds < 60)
                sec = seconds;
        else
                sec = 0;
}

clockType::clockType()  //default constructor
{
        hr = 0;
        min = 0;
        sec = 0;
}

//class5.cpp
#include <iostream>
#include "class5.h"

using namespace std;

int main()
{
        clockType myClock;  //default constructor
        clockType yourClock (5,12,40);                        //Line 1
        int hours;
        int minutes;
        int seconds;

   cout<<"Line 2: myClock: ";                                       //Line 2
        myClock.printTime();                                        //Line 3
        cout<<endl;                                                         //Line 4

        cout<<"Line 5: yourClock: ";                          //Line 5
        yourClock.printTime();                                              //Line 6
        cout<<endl;                                                         //Line 7
        yourClock.setTime(5,45,16);                                   //Line 8

        cout<<"Line 9: After setting - yourClock: ";        //Line 9
        yourClock.printTime();                                        //Line 10
        cout<<endl;                                                   //Line 11

        if(myClock.equalTime(yourClock))                          //Line 12
          cout<<"Line 13: Both times are equal."
                <<endl;                                               //Line 13
        else                                                          //Line 14
          cout<<"Line 15: The two times are not equal"
                <<endl;                                               //Line 15

        cout<<"Line 16: Enter hours, minutes, and "
           <<"seconds: ";                                             //Line 16
        cin>>hours>>minutes>>seconds;                             //Line 17
        cout<<endl;                                                   //Line 18

        myClock.setTime(hours,minutes,seconds);                   //Line 19

        cout<<"Line 20: New myClock: ";                           //Line 20
        myClock.printTime();                                          //Line 21
        cout<<endl;                                                   //Line 22

        myClock.incrementSeconds();                                   //Line 23

        cout<<"Line 24: After incrementing the clock by "
               <<"one second, myClock: ";                             //Line 24
        myClock.printTime();                                          //Line 25
    cout<<endl;                                                       //Line 26
        return 0;
}//end main
```

output

```
Line 2: myClock: 00:00:00
Line 5: yourClock: 05:12:40
Line 9: After setting - yourClock: 05:45:16
Line 15: The two times are not equal
Line 16: Enter hours, minutes, and seconds: 12 24 28

Line 20: New myClock: 12:24:28
Line 24: After incrementing the clock by one second, myClock: 12:24:29
```

```
//  SPECIFICATION FILE          ( class6.h )
```

```
class  TimeType     // declares a  class data type
{                                                   //  does not allocate memory

public :                          //  5 public function members

        void         Set ( int  hours , int  mins , int  secs ) ;
        void          Increment ( ) ;
        void          Write ( )  const ;
        void                   PrintAll (TimeType times[], int numberOfTimes ) const;
        bool         Equal ( TimeType   otherTime )  const ;
        bool         LessThan ( TimeType   otherTime )  const ;
        TimeType ( int  initHrs ,  int  initMins ,  int  initSecs ) ; // constructor

        TimeType ( ) ;                                          // default constructor


private :                                 //  3 private data members

        int          hrs ;
        int          mins ;
        int            secs ;
} ;
//   IMPLEMENTATION FILE                  ( class6i.cpp )
//   Implements the TimeType member functions.

#include <iostream>
using namespace std;
#include "class6.h"  // also must appear in client code

// private data members
//                int hrs ;
//                int mins ;
//                int secs ;

void  TimeType::Set(int hours, int minutes, int seconds)
        {
                hrs = hours ;
                mins = minutes ;
                secs = seconds ;
        }

void  TimeType::Increment()
        {
                secs++ ;
                if (secs > 59)
                        {
                                secs = 0;
                                mins++;
                                if (mins > 59)
                                        {
                                                mins = 0;
                                                hrs++;
                                                if (hrs > 23)
                                                        hrs=0;
                                        }
                        }
        }

void   TimeType :: Write ( )   const

    //   Postcondition:   Time has been output in form HH:MM:SS

        {       if  ( hrs < 10 )
                                cout << '0' ;
                cout  << hrs  <<  ':' ;
                if  ( mins < 10 )
                                cout << '0' ;
                cout  << mins  <<  ':' ;
                if  ( secs < 10 )
                                cout << '0' ;
                cout  << secs << endl;
        }

        void   TimeType :: PrintAll (TimeType times[],int numberOfTimes )   const

    //   Postcondition:   Time has been output in form HH:MM:SS

        {
                cout << "entering PrintAll" << endl;
                for (int index=0; index < numberOfTimes; index ++)
                {
                if  (times[index].hrs < 10 )
                                cout << '0' ;
                cout  << times[index].hrs  <<  ':' ;
                if  ( times[index].mins < 10 )
                                cout << '0' ;
                cout  << times[index].mins  <<  ':' ;
                if  ( times[index].secs < 10 )
                                cout << '0' ;
                cout  << times[index].secs ;
                cout  << endl;
                }
        }


bool  TimeType :: Equal ( /* in */  TimeType otherTime ) const
  //  Postcondition:
  //      Function value == true,    if this time equals otherTime
  //                                 == false , otherwise
  {
                return ( (hrs == otherTime.hrs)  && (mins == otherTime.mins)
                                                        && (secs  == otherTime.secs) ) ;
        }

bool  TimeType :: LessThan ( /* in */  TimeType otherTime ) const

  {
        return ((hrs < otherTime.hrs)   ||
                        (hrs == otherTime.hrs && mins < otherTime.mins) ||
                        (hrs == otherTime.hrs && mins == otherTime.mins
                         &&secs < otherTime.secs));

  }

TimeType :: TimeType (  )
//  Default  Constructor
//  Postcondition:
//                      hrs == 0   &&   mins == 0   &&  secs == 0
{
                hrs  =  0 ;
      mins = 0 ;
                secs = 0 ;
}

TimeType :: TimeType ( /* in */   int   initHrs,
                                /* in */   int   initMins,
                                /* in */   int   initSecs )
//  Constructor
//  Precondition:  0 <= initHrs <= 23    &&    0 <= initMins <= 59
//                       0 <= initSecs <= 59
//  Postcondition:
//                 hrs == initHrs  &&  mins == initMins  && secs == initSecs
{
                 hrs  =  initHrs ;
       mins =  initMins ;
       secs =  initSecs ;
}
// class6.cpp
// class6.cpp
// to illustrate creating a using an array of class objects
//
        #include <iostream>
using namespace std;

        #include "class6.h"

int main()
{
        TimeType startTime[2];   // create two instances of a class
        startTime[1]=TimeType(6,3,1);  //initializes an array object within an array
                                                                // a temporary object is
created
        startTime[1].Write();

        startTime[0].Write();
        startTime[0].Set(10,20,30) ;
        startTime[0].Write();


        startTime[1].Set(12,24,49) ;
        startTime[1].Write();


        startTime[0].PrintAll(startTime,2);

        if (startTime[0].Equal(startTime[1]))
                cout << "times are equal" << endl;
        else
                cout << "times are not equal" << endl;
 }
```
output

```
06:03:01
00:00:00
10:20:30
12:24:49
entering PrintAll
10:20:30
12:24:49
times are not equal
```

```
 // class7.cpp
// conversions: Distance to meters, meters to Distance
#include <iostream>
```

```
using namespace std;

const double FEETTOMETERS = 1.0/3.280833;       // feet to meters

class Distance                     // English Distance class
          {

          public:
                    Distance()  ;                 // constructor (no args)
                    Distance( double in ) ;       // constructor (one arg)
                    Distance(int ft, double in);  // constructor (two args)

          void getDist( );             // get length from user

          void showDist( ) ;           // display distance

          private:
                    int feet;
                    double inches;
          };
            // implementation file

          Distance :: Distance()                 //  default constructor (no args)
           {
           feet = 0; inches = 0.0;
           }

          Distance ::Distance(  double in  )    // constructor (one arg)
           {
                              feet = int(in/12);   // number of feet
                              inches = int(in)%12; // number of inches
            }

            Distance ::Distance(int ft, double in)  // constructor (two args)
           {
                    feet = ft;
                    inches = in;
           }

                    void Distance ::getDist( )             // get length from user
           {
           cout << "\nEnter feet: ";  cin >> feet;
           cout << "Enter inches: ";  cin >> inches;
           }

                    void Distance ::showDist( )            // display distance
```

```
	{
		cout << feet << "\"" << inches << '\"';
		double meters;
		meters = feet * FEETTOMETERS + (inches / 12.0) * FEETTOMETERS ;
		cout << " is "  << meters << " meters" << endl;
	}

	int main()
	{
	  int index ;

	  Distance myDistance;
	  myDistance.showDist();
	  myDistance.getDist();
	  myDistance.showDist();

	  Distance yourDistance(12);
	  yourDistance.showDist();

	  Distance ourDistance(2,15);
	  ourDistance.showDist();

	  // array of distances
	  Distance arrayDistance[3];

	  for (index=0;index < 3; index++)
	  {
				arrayDistance[index].getDist();
				arrayDistance[index].showDist();
	  }
	}
```

output

```
0'0" is 0 meters

Enter feet: 12
Enter inches: 33
12'33" is 4.49581 meters
1'0" is 0.304801 meters
2'15" is 0.990602 meters

Enter feet: 33
Enter inches: 11
33'11" is 10.3378 meters
```

```
Enter feet: 44
Enter inches: 6
44'6" is 13.5636 meters

Enter feet: 22
Enter inches: 77
22'77" is 8.66142 meters
```

```cpp
// class8.cpp
// static class data
#include <iostream>
using namespace std;


class staticClass
        {
  private:
    static int count;   // only one data item for all objects
                        // note: *declaration* only!
                                int data;
        public:
                staticClass();
                staticClass(int input);
                void getcount();
        };

                staticClass::staticClass()
                {
                 count++;
                 data=count;
                }

                staticClass::staticClass(int input)
                {
                count++;
                data=input;
                 }

                void   staticClass::getcount()
                 {
                 cout << "data value is " << data;
                 cout << " count is " << count << endl;
                 }

        class dynamicClass
  {
```

```
  private:
                int count;
                int data;
  public:
                dynamicClass();
                void getcount();
        };

                dynamicClass::dynamicClass()
                {
                count=0;
                count++;
    data=count;
                }

                void dynamicClass::getcount()
                {
                 cout << "data value is " << data;
                 cout << " count is " << count << endl;
                 }
int staticClass::count;         // *definition* of count


int main()
        {
        staticClass staticObject1, staticObject2, staticObject3;

        staticObject1.getcount();  // each object
        staticObject2.getcount();  // sees the same
        staticObject3.getcount();  // value of count

        staticClass staticObject4(30);
        staticObject4.getcount();

        dynamicClass dynamicObject1, dynamicObject2, dynamicObject3;
        dynamicObject1.getcount();
        dynamicObject2.getcount();
        dynamicObject3.getcount();
        }
```
output

```
data value is 1 count is 3
data value is 2 count is 3
data value is 3 count is 3
data value is 30 count is 4
data value is 1 count is 1
data value is 1 count is 1
```

data value is 1 count is 1

```
// class9.h
class  MyTime        // declares a  class data type
{                                                  //  does not allocate memory

public :                          //  5 public function members

        void        Set ( int  hours , int  mins , int  secs ) ;
        void         Increment ( ) ;
        void         Write ( )  const ;
        bool        Equal ( const MyTime &  otherTime ) ;
        bool        LessThan ( MyTime   otherTime )  const ;

private :                               //  3 private data members

        int          hrs ;
        int          mins ;
        int            secs ;
} ;


// class9i.cpp

//   IMPLEMENTATION FILE
//   Implements the MyTime member functions

#include <iostream>
using namespace std;
#include "class9.h"  // also must appear in client code

// private data members
//                int hrs ;
//                int mins ;
//                int secs ;

void  MyTime::Set(int hours, int minutes, int seconds)
        {
                hrs = hours ;
                mins = minutes ;
                secs = seconds ;
        }

void  MyTime::Increment()
        {
                secs++ ;
                if (secs > 59)
                        {
                                secs = 0;
                                mins++;
                                if (mins > 59)
                                        {
                                                mins = 0;
                                                hrs++;
                                                if (hrs > 23)
                                                        hrs=0;
                                        }
                        }
        }

void   MyTime :: Write ( )   const

          //   Postcondition:   Time has been output in form HH:MM:SS

        {       if  ( hrs < 10 )
                                cout << '0' ;
                cout  << hrs  <<  ':' ;
                if  ( mins < 10 )
                                cout << '0' ;
                cout  << mins  <<  ':' ;
                if  ( secs < 10 )
                                cout << '0' ;
                cout  << secs ;
         //     mins++; // violates const parameter
        }

bool  MyTime :: Equal ( /* in */const  MyTime& otherTime )
  //  Postcondition:
  //      Function value == true,    if this time equals otherTime
  //                                 == false , otherwise
  {
                 return ( (hrs == otherTime.hrs)  && (mins == otherTime.mins)
                                                         && (secs  == otherTime.secs) ) ;
         //     otherTime.mins++; // violates const
                mins++; //this is valid
                mins-- ;        // this is valid
        }

bool  MyTime :: LessThan ( /* in */  MyTime otherTime ) const

  {
        return ((hrs < otherTime.hrs)   ||
                                    (hrs == otherTime.hrs && mins < otherTime.mins) ||
                                    (hrs == otherTime.hrs && mins == otherTime.mins
                                     &&secs < otherTime.secs));
                          //        mins++;  violates const parameter
                          otherTime.mins++; // valid
                          otherTime.mins--; // valid
  }
// to illustrate creating and using classes
//
        #include <iostream>
using namespace std;

        #include "class9.h"

int main()
{
        MyTime startTime;   // create an instance of a class
        startTime.Set(10,20,30) ;
        startTime.Write();
        cout << endl;

        MyTime endTime;   // create an instance of a class
        endTime.Set(12,24,49) ;
        endTime.Write();
        cout << endl;

        endTime.Increment();
        endTime.Write();
        cout << endl;

        if (startTime.Equal(endTime))
                cout << "times are equal" << endl;
        else
                cout << "times are not equal" << endl;
}
```

output

```
10:20:30
12:24:49
12:24:50
times are not equal
```

```
//class10.h
#include <string>
using namespace std;

class personType
```

```
{
public:
   void print() const;
   void setName(string first, string middle, string last);
         void setLastName(string last);
         void setFirstName(string first);
         void setMiddleName(string middle);

         bool isLastName(string last);
         bool isFirstName(string first);

   void getName(string& first, string& middle, string& last);

   personType(string first, string middle, string last);

   personType();

 private:
   string firstName;
         string middleName;
   string lastName;
};
//class10i.cpp
//class10i.cpp
#include <iostream>
#include <string>
#include "class10.h"

using namespace std;

void personType::print() const
{
         cout<<firstName<<" "<<middleName<<" "<<lastName;
}

void personType::setName(string first, string middle, string last)
{
         firstName = first;
         middleName = middle;
         lastName = last;
}

void personType::setLastName(string last)
{
         lastName = last;
}
```

```
void personType::setFirstName(string first)
{
	firstName = first;
}

void personType::setMiddleName(string middle)
{
	middleName = middle;
}

bool personType::isLastName(string last)
{
	return (lastName == last);
}

bool personType::isFirstName(string first)
{
	return (firstName == first);
}


void personType::getName(string& first, string& middle, string& last)
{
	first = firstName;
	middle = middleName;
	last = lastName;
}

//constructor with parameters
personType::personType(string first, string middle, string last)

{
	firstName = first;
	middleName = middle;
	lastName = last;
}

personType::personType()   //default constructor
{
	firstName = "";
	middleName = "";
	lastName = "";
}
//class10.cpp
```

```
#include <iostream>
#include "class10.h"
using namespace std;

int main()
{
	personType student("Mary", "Beth", "Jones");

	student.print();
	cout<<endl;

	if(student.isLastName("Regan"))
		cout<<"Student\'s last name is Regan"<<endl;
	else
		cout<<"Student\'s last name is not Regan"<<endl;

	return 0;
}
```

output

```
Mary Beth Jones
Student's last name is not Regan
```