CALIFORNIA STATE UNIVERSITY, SACRAMENTO

College of Business Administration

MIS 211 - Information Systems II

C++ Solution to Homework 4

Main Function

```
#include <fstream.h>
#include <iostream.h>
#include <stdlib.h>
#include <iomanip.h>
#include <stdio.h>
#include <time.h>
#include <string.h>
#include <direct.h>
//
class videos                    ← Class declaration
{
       protected:
              double price[100], stock_num[100];
              int i, num_compares, total_compares;
              short j;

       public:
              void AssignVal(double, double, int);
              double PricePeek(short);
};

void videos::AssignVal(double a, double b, int i)
{
       stock_num[i] = a;
       price[i] = b;
}

//--------------------------- Price Peek ---------------------------
double videos::PricePeek(short i)
{
       double v_price;

       v_price = price[i];

       return v_price;
}
```

```
//=========================================================================
//  SEQUENTIAL SEARCH SUBCLASS
//
class seq_search : public videos                    ← Sequential search subclass declaration
{
protected:

       int sub, i, total_compares, compares;
       short found;

public:
       seq_search();
       int search_array(double, int);
       short found_peek();
       int count_peek();
};

seq_search::seq_search()                            ← Constructor
{
       total_compares = 0;
}
```

Sequential Search Method

```
int seq_search::search_array (double search_stock_num, int elements)
{
       compares = 1;
       found = 100;                                 ← Set 'found' switch to an (absurd) initial
//                                                      value (0 = not found, 1 = found)
       for (i=0; i <= elements && found > 1; i+=1)
       {
              if (stock_num[i] == search_stock_num)
                     found = 1;
              else
              {      compares += 1;
                     if (stock_num[i] > search_stock_num)      ← Quick exit
                     {      found = 0;
                            compares -= 1;
                     }
              }
//
       }
       if (i > elements)
              found = 0;
//
       total_compares += compares;
//
       return compares;
};

short seq_search::found_peek()
{
       return found;
}
```

```
int seq_search::count_peek()
{
       return total_compares;
}

//=====================================================================
//  BINARY SEARCH SUBCLASS
//
class bin_search : public videos
{
protected:
       int sub, compares, total_compares;
       short found, lower, mid, upper;

public:
       bin_search();
       short search_array (double, int);
       short found_peek ();
       int count_peek ();
};

bin_search::bin_search()
{
       total_compares = 0;
}
```

← Binary search class declaration

← Constructor

Binary Search Method

```
short bin_search::search_array (double search_stock_num, int elements)
{
       compares = 0;
       found = 100;                                  ← Set 'found' switch to an initial value
//                                                         (0 = not found, 1 = found)
       lower = 0;                                    ← Set initial subscript locations
       upper = elements - 1;
       mid =  (elements / 2) - 1;

       while (found > 1)
       {
              i = mid;
              compares += 1;

              if (stock_num[i] == search_stock_num)
                     found = 1;
              else
              {
                     if (stock_num[i] > search_stock_num)
                     {
                            upper = mid;
                            mid = ((mid - lower) / 2) + lower;
                     }
                     else  // (stock_num[i] < search_stock_num)
                     {
                            lower = mid;
                            mid = ((upper - mid) / 2) + mid;
                     }
                     if (mid - lower < 1 || upper - mid < 1)        ← Not found
                            found = 0;
              }
       }
//
       total_compares += compares;

       return compares;
}

short bin_search::found_peek ()
{
       return found;
}

int bin_search::count_peek ()
{
       return total_compares;
}
```

```
//---------------------------------------------------------------------
int main()                                          ← Main function
{
       char  vid_title[41], vid_title_array[100][41];
       double stk_num = 0, price = 0, s_num = 0;
       short count = 0, i = 0,
              seq_results = 0, bin_results = 0,
              bfound = 0, sfound = 0,
              stock_num_count = 0,
              elements = 0;
       int total_seq_compares = 0,
              total_bin_compares = 0;

       double test_double = 99;

       seq_search seq_object;                       ← Instantiate sequential search object
       bin_search bin_object;                       ← Instantiate binary search object

       ifstream infile;
       infile.open("videos.dat");

//     Records from the videos file are sequentially loaded into the array

       infile >> stk_num;
       infile.getline(vid_title,41);
       infile >> price;

       while (!infile.eof())                        ← Begin loop to load arrays
       {
              strcpy(vid_title_array[count],vid_title);     ← Load the arrays
              bin_object.AssignVal(stk_num,price,count);
              seq_object.AssignVal(stk_num,price,count);
              count += 1;

              infile >> stk_num;
              infile.getline(vid_title,41);
              infile >> price;
       }                                            ← End loop

       cout << count << "\n";

       ifstream numfile;
       ofstream output;

       numfile.open("stocknum.dat");                ← Open stocknum.dat file for input
       output.open("report.txt");
```

```
//     Records from the stocknum file are sequentially read

       numfile >> stk_num;                ← Priming read of the stocknum.dat file

       while (!numfile.eof())             ← Begin loop to search stock number array
       {
              stock_num_count += 1;
//
              seq_results = seq_object.search_array(stk_num, count);     ← Sequential
              sfound = seq_object.found_peek();                            Search
//
              bin_results = bin_object.search_array(stk_num, count);     ← Binary
              bfound = bin_object.found_peek();                            Search
//
              if (sfound == 0)
              {
                     output << setw(15) << "Video Not Found"          ← Print output
                                   << setw(8)    << stk_num << " "
                                   << setw(40) << setfill(' ')
                                   << setw(55)   << seq_results
                                   << setw(5)    << bin_results
                                   << "\n";
              }
              else
              {
                     output << setw(15)   << "Video Found    " ← Print output
                                   << setw(8)    << stk_num << " "
                                   << setw(40)
                                   << vid_title_array[seq_results-1]
                                   << setw(10)
                                   << seq_object.PricePeek(seq_results-1)
                                   << setw(5)    << seq_results
                                   << setw(5)    << bin_results
                                   << "\n";
              }
              numfile >> stk_num;         ← Second read of the stocknum.dat file
       }                                  ← End loop

       output  << "\n\n" << setw(58) << " "      ← Print report footing
                     << "Total compares: "
                     << setw(5)    << seq_object.count_peek()
                     << setw(5)    << bin_object.count_peek()
                     << "\n";

       output << setw(40)   <<" "
                     << "Average compares per search: "
                     << setw(5)
                     << seq_object.count_peek()/stock_num_count
                     << setw(5)
                     << bin_object.count_peek()/stock_num_count
                     << "\n";

       infile.close();
```

```
        numfile.close();
        output.close();
        return 0;
}
```