```
//---------------------------------------------------------------
//
//  Record.h
//  p5
//
//---------------------------------------------------------------


#ifndef __p5__Record__
#define __p5__Record__

#include <stdio.h>
#include <iostream>
#include <cstdlib>
#include <fstream>
#include <iomanip>

using namespace std;

class Record{
public:
    int key;
    int offset;
    Record();
    Record(int key,int offset);
    int getKey() const;
    int getOffset() const;
    double getCost() const;
    void printDetails();
    bool operator <(const Record& r2)const;
    bool operator ==(const Record& r2)const;
    bool operator> (const Record& r2)const;

private:

    void setKey(int n);
    void setOffset(int n);

};

#endif /* defined(__p5__Record__) */
```

```
printf \\n

cat -b Record.cpp
```

```
//--------------------------------------------------------------
//
//  Item.cpp
//  p4
//
//-------------------------------------------------------------

#include "Record.h"
Record::Record(){
    //-------------------------------------------------------------
    // Default Item Constructor
    //-------------------------------------------------------------

    setKey(0);
    setOffset(0);
}

Record::Record(int key, int offset){
    //--------------------------------------------------------------------
    //Preconditions: 3 integers and a string are passed to the constructor
    //               from the calling code
    //Postconditions: an Item object is instantiated with the passed values
    //
    //Variables used: stock - integger representing an item's stock number
    //                 desc - description of an item, in a string
    //                  qty - integer value with the current quantity
    //             nextItem - next item in the stock list
    //--------------------------------------------------------------------

    setKey(key);
    setOffset(offset);
}

void Record::printDetails(){
    cout << setw(6)<< getKey()<<setw(2)<< getOffset();
    cout<<endl;
    cout.clear();
}

//---------------------------------------------------------------------
//BEGIN GETTERS AND SETTERS
//---------------------------------------------------------------------

int Record::getKey()const{
    return key;
}

int Record::getOffset()const{
    return offset;
}


void Record::setKey(int stock){
    key = stock;
}


void Record::setOffset(int n){
    offset = n;
}

//---------------------------------------------------------------------
//END GETTERS AND SETTERS
//---------------------------------------------------------------------

bool Record::operator< (const Record& r2)const{
    return (key < r2.getKey());
}

bool Record::operator> (const Record& r2)const{
    return (key > r2.getKey());
}

bool Record::operator ==(const Record& r2)const{
    return(key == r2.getKey());
}
printf \\n\\n


cat -b Item.h
//--------------------------------------------------------------
//
//  Item.h
//  p4
//
//--------------------------------------------------------------


#ifndef __p4__item__
```

```
#define __p4__item__

#include <stdio.h>
#include <iostream>
#include <cstdlib>
#include <fstream>

using namespace std;

class Item{
public:
    Item();
    Item(int stockNum, string description, int qty, double cost);
    int getStockNum() const;
    string getDescription() const;
    int getCount() const;
    double getCost() const;

private:
    int stockNum;
    char description[8];
    int count;
    double cost;
    void setStockNum(int n);
    void setDescription(string str);
    void setCount(int n);
    void setCost(double n);
};


#endif /* defined(__p4__item__) */
```

```
printf \\n

cat -b Item.cpp
```

```
//--------------------------------------------------------------
//
//  Item.cpp
//  p4
//
//--------------------------------------------------------------


#include "Item.h"
```

```
Item::Item(){
    //------------------------------------------------------------
    // Default Item Constructor
    //------------------------------------------------------------

    setStockNum(0);
    setDescription("");
    setCount(0);
    setCost(0);
}

Item::Item(int stock, string desc, int qty, double c){
    //-------------------------------------------------------------------
    //Preconditions: 3 integers and a string are passed to the constructor
    //               from the calling code
    //Postconditions: an Item object is instantiated with the passed values
    //
    //Variables used: stock - integger representing an item's stock number
    //                  desc - description of an item, in a string
    //                   qty - integer value with the current quantity
    //              nextItem - next item in the stock list
    //-------------------------------------------------------------------

    setStockNum(stock);
    setDescription(desc);
    setCount(qty);
    setCost(c);
}

//-------------------------------------------------------------------
//BEGIN GETTERS AND SETTERS
//-------------------------------------------------------------------

int Item::getStockNum()const{
    return stockNum;
}

string Item::getDescription()const{
    return description;
}

int Item::getCount()const{
    return count;
}
```

```
    47	double Item::getCost()const{
    48	    return cost;
    49	}

    50	void Item::setStockNum(int stock){
    51	    stockNum = stock;
    52	}

    53	void Item::setDescription(string desc){
    54	    for (int i = 0; i<8;i++){
    55	        description[i]=desc[i];
    56	    }
    57	
    58	}

    59	void Item::setCount(int qty){
    60	    count = qty;
    61	}

    62	void Item::setCost(double c){
    63	    cost = c;
    64	}

    65	//---------------------------------------------------------------------
    66	//END GETTERS AND SETTERS
    67	//---------------------------------------------------------------------

printf \\n\\n


cat -b CreateIndex.h
     1	//--------------------------------------------------------------
     2	//
     3	//  CreateIndex.h
     4	//  p5
     5	//
     6	//--------------------------------------------------------------


     7	#ifndef p5_CreateIndex_h
     8	#define p5_CreateIndex_h
     9	#include <list>
```

```
class CreateIndex{
public:

    CreateIndex();
    void run();
    void printInventory(list <Record> myList, fstream& myFile);
    int getRecords();
    void createBinaryFile(int numRecords);
private:
    int records;
};
#endif
printf \\n

cat -b CreateIndex.cpp
//---------------------------------------------------------------
//
//  main.cpp
//  p5
//
//---------------------------------------------------------------

#include "Item.h"
#include "Record.h"
#include "CreateIndex.h"

CreateIndex::CreateIndex(){
    //---------------------------------------------------------------
    // DEFAULT CONSTRUCTOR
    //---------------------------------------------------------------
}

void CreateIndex::run() {

    //---------------------------------------------------------------
    //Preconditions: The calling code has called this method
    //
    //Postconditions: An index file containing
    //
    //Variables used: infile1: fstream object that accesses prog5.idx
    //                infile2: fstream object to the data file prog5.dat
    //                numRecords: number of records in the data file
    //                keyArr[]: an integer array used to store keys
    //                item: used to reference a Record object
    //                bsearchResult: integer containing the result
```

```
        //                   of the binarySearchMethod
        //                 stockNum: int to hold the Stock value
        //                 description: string description of the current record
        //                 count: count value for the current record
        //                 cost: double value of the cost of the current record
        //
        //----------------------------------------------------------------

        list <Record> myList;
        Record record;
        Item item;
        int lines = 0;

        //open fstream of input file


        fstream infile1("../instr/prog5.dat", ios::in);
        if (!infile1.is_open()) {
            cout<<"could not open prog5.dat"<<endl;
        }else{
        int numRecords;
        infile1>>numRecords;
        records = numRecords;
        infile1.close();
        list <int> myList1;

        fstream infile2("../instr/prog5.dat", ios::in);


        createBinaryFile(numRecords);


        fstream myFile("prog5bin.dat", ios::in|ios::out|ios::binary);


        int stockNum;
        string description;
        int count;
        double cost;
        int n=0;


        while (infile2 >> stockNum >> description >> count >> cost) {



            item = Item(stockNum, description, count, cost);
            //Creates Item objects and writes the information contained within the object to the output file
            record = Record(stockNum, lines);

            //following line moves the filepointer to the item's location in bytes on the list
            myFile.seekg(lines*sizeof(Item));

            //following line writes the information to the output file
            myFile.write(reinterpret_cast<const char *>(&item), sizeof(Item));



            myList1.push_back(stockNum);

            myList.push_back(record);

            n++;
            lines++;
        }

        infile2.close();//close input file


        //following block of code creates the output file
        ofstream openFile;
        openFile.open("prog5.idx", ios::out);
        openFile.close();
        //end creation of output file

        fstream indexFile("prog5.idx", ios::out);
        myList.sort();
        printInventory(myList, indexFile);

        cout<<endl;

        }


    }

    void CreateIndex::printInventory(list <Record> myList, fstream& myFile){
        //-----------------------------------------------------------------
        //Preconditions: a reference to an output stream and a reference
        //               to List were passed by the calling code
        //Postconditions: the contents of the list are sent to
        //                the output stream
        //
        //Variables used: &myfile: reference to an output stream
        //                item: reference to an Item object
        //----------------------------------------------------------------

        for(std::list<Record>::iterator it = myList.begin(); it!= myList.end(); ++it)
        {
            Record item = *it;
            if(item.getOffset()>0){
            myFile <<left << setw(6)<< item.getKey()<<setw(2)<< item.getOffset()<<endl;
            }
        }
    }

    int CreateIndex::getRecords(){
        return records;
    }

    void CreateIndex::createBinaryFile(int numRecords){
        //----------------------------------------------------------------
        //Preconditions: number of records in the data file is passed by
        //                  the calling code
        //Postconditions: a binary file is created that has the same
        //                information as the data file
        //
        //Variables used: openfile: ofstream used to create the binary file
        //                item: reference to an Item object
        //----------------------------------------------------------------

        ofstream openFile;
        openFile.open("prog5bin.dat", ios::out|ios::binary);
        openFile.close();

        Item item = Item();
        fstream myFile("prog5bin.dat", ios::in|ios::binary);

        int i = 0;
        for(i = 0; i<numRecords+1;i++){
            myFile.write(reinterpret_cast< const char * >(&item), sizeof(Item));
        }
        myFile.close();
    }
```

```
printf \\n\\n


cat -b SearchIndex.h
//--------------------------------------------------------------
//
//  SearchIndex.h
//  p5
//
//--------------------------------------------------------------

#ifndef __p5__SearchIndex__
#define __p5__SearchIndex__

#include "Item.h"
#include "Record.h"
#include <sstream>
#include <list>

using namespace std;

class SearchIndex{
public:
    SearchIndex();
    void run(string searchKey, string fileName, int records);
    int binarySearch(int keyArr[], int numRecords, int key);
    void getRecord(int offset);
    void outputLine(const Item &record);
    int convert(const string& str);
};

#endif /* defined(__p5__SearchIndex__) */
printf \\n

cat -b SearchIndex.cpp
//--------------------------------------------------------------
//
//  SearchIndex.cpp
//  p5
//
//--------------------------------------------------------------

#include "Record.h"
```

```
#include "Item.h"
#include "SearchIndex.h"

SearchIndex::SearchIndex(){
    //----------------------------------------------------------------
    // DEFAULT CONSTRUCTOR
    //----------------------------------------------------------------
}

void SearchIndex::run(string searchKey, string fileName, int records){
    //----------------------------------------------------------------
    //Preconditions: A key string values, a string containing a filename
    //                of an index file, and the total number of records
    //
    //Postconditions: The integer value of the search key is returned
    //               if found, else 0 is returned to the calling code
    //
    //Variables used: myFile: fstream object that accesses prog5.idx
    //                infile: fstream object to the data file prog5.dat
    //                sKey: integer value of the search key
    //                offset: integer containing the RRN (offset)
    //                keyArr[]: an integer array used to store keys
    //                item: used to reference a Record object
    //                bsearchResult: integer containing the result
    //                  of the binarySearchMethod
    //
    //----------------------------------------------------------------

    fstream myFile(fileName.c_str(), ios::in);
    fstream infile("../instr/prog5.dat", ios::in);

    int sKey;
    sKey = convert(searchKey);
    int n=0;
    int key, offset;

    int keyArr[records];

    int i = 0;

    Record item;

    while (myFile >> key >> offset) {
        while(i!=offset){
            i++;
        }

        //Creates Item objects and writes the information contained within the object to the output file
        keyArr[i] = key;
        i=0;
        n++;

    }

    int bSearchResult = binarySearch(keyArr, records, sKey);

    if (bSearchResult > 0) {
        getRecord(bSearchResult);

    } else {
        cout << "Key not found, please try again." << endl;
    }



}

int SearchIndex::binarySearch(int keyArr[], int numRecords, int key){
    //-------------------------------------------------------------
    //Preconditions: An array of key values, a total number of records
    //               and a key to be search for is passed to the method
    //
    //Postconditions: The integer value of the search key is returned
    //               if found, else 0 is returned to the calling code
    //
    //Variables used: mid: middle index within the array to be compared
    //                lower: leftmost index of the array to be compared
    //                upper: rightmost  index of the array to be compared
    //-------------------------------------------------------------

    int mid,lower = 0;
    int upper = numRecords;
    while( lower <= upper )
    {
        mid = ( lower + upper )/2;
        if( key > keyArr[mid] )
            lower = mid+1;
        else if(key < keyArr[mid])
            upper = mid-1;
        else
            return mid;
    }
    return 0;
}

void SearchIndex::getRecord(int offset){
    //----------------------------------------------------------------
    //Preconditions: An integer is passed to the method
    //
    //Postconditions: The record referenced by the passed integer is
    //                printed to the screen
    //
    //Variables used: inputFile: reference to an input file stream
    //                &item reference to an Item object
    //----------------------------------------------------------------

    Item item;

    fstream inputFile("prog5bin.dat", ios::in|ios::binary);
    inputFile.seekg(offset*sizeof(item));

    inputFile.read(reinterpret_cast<char * >(&item), sizeof(Item));
    outputLine(item);
    inputFile.close();
}

void SearchIndex::outputLine(const Item &record){
    //----------------------------------------------------------------
    //Preconditions: a reference to an output stream and a reference
    //               to an Item object were passed by the calling code
    //Postconditions: the contents of the record object are sent to
    //                the output stream
    //
    //Variables used: &output: reference to an output stream
    //                &record reference to an Item object
    //----------------------------------------------------------------
    cout << right << setw(5)<< record.getStockNum()<<right <<setw(8) << record.getDescription() << right << setw(3)<< record.getCount()<< right<<setw(6)<<record.getCost()<<endl;

    cout.clear();


}

int SearchIndex::convert(const string& str) {
    stringstream ss(str);
    int n;
    ss >> n;
    return n;
}

printf \\n\\n


cat -b p5.cpp
/*
 PROGRAM NAME: Program 5: Indexed Files

 PROGRAMMER:   James Francis

 CLASS:        CSC 331.001, Fall 2014

 INSTRUCTOR:   Dr. Robert Strader

 DATE STARTED: October 21, 2014

 DUE DATE:     October 28, 2014

 PROGRAM PURPOSE:

 1) Create a binary file from prog5.dat
 2) Read in records from prog5.dat to Record objects to a list
 3) Write the Item objects to the previously created binary file
 4) Print to console the item record in the file going from one
 item to the next based on their nextItem variable


 VARIABLE DICTIONARY:

 indexer: CreateIndex object that will create an index of records within the data file
 searcher: SearchIndex object that will perform search operations
 key: String holds the value of the key to be searched for
 fileName: String holds the value of the index file

 ADTs: std::list

 FILES USED: prog5.dat


 SAMPLE INPUTS:
search 12382 prog5.idx


 SAMPLE OUTPUTS:

 12382 Item09 62 41.37


 ---------------------------------------------------------------------*/

#include "Record.h"
#include "Item.h"
#include "CreateIndex.h"
#include "SearchIndex.h"

int main(int argc, const char * argv[]){

    CreateIndex indexer = CreateIndex();
    indexer.run();

    SearchIndex searcher = SearchIndex();
    string key = argv[1];
    string fileName = argv[2];

    searcher.run(key,fileName, indexer.getRecords());

    return 0;
}
```