

```

1  //-----
2  //
3  //  Record.h
4  //  p5
5  //
6  //-----

7  #ifndef __p5__Record__
8  #define __p5__Record__

9  #include <stdio.h>
10 #include <iostream>
11 #include <cstdlib>
12 #include <fstream>
13 #include <iomanip>

14 using namespace std;

15 class Record{
16 public:
17     int key;
18     int offset;
19     Record();
20     Record(int key,int offset);
21     int getKey() const;
22     int getOffset() const;
23     double getCost() const;
24     void printDetails();
25     bool operator <(const Record& r2)const;
26     bool operator ==(const Record& r2)const;
27     bool operator > (const Record& r2)const;
28
29 private:
30     void setKey(int n);
31     void setOffset(int n);

32 };

33 #endif /* defined(__p5__Record__) */

```

printf \\n

cat -b Record.cpp

```

1  //-----
2  //
3  //  Item.cpp
4  //  p4
5  //
6  //-----

7  #include "Record.h"
8  Record::Record(){
9      //-----
10     // Default Item Constructor
11     //-----
12
13     setKey(0);
14     setOffset(0);
15 }

16 Record::Record(int key, int offset){
17     //-----
18     //Preconditions: 3 integers and a string are passed to the constructor
19     //                from the calling code
20     //Postconditions: an Item object is instantiated with the passed values
21     //
22     //Variables used: stock - integer representing an item's stock number
23     //                desc - description of an item, in a string
24     //                qty - integer value with the current quantity
25     //                nextItem - next item in the stock list
26     //-----
27
28     setKey(key);
29     setOffset(offset);
30 }

31 void Record::printDetails(){
32     cout << setw(6)<< getKey()<<setw(2)<< getOffset();
33     cout<<endl;
34     cout.clear();
35 }

36 //-----
37 //BEGIN GETTERS AND SETTERS
38 //-----

39 int Record::getKey()const{

```

```

40     return key;
41 }

42 int Record::getOffset()const{
43     return offset;
44 }

45 void Record::setKey(int stock){
46     key = stock;
47 }

48 void Record::setOffset(int n){
49     offset = n;
50 }

51 //-----
52 //END GETTERS AND SETTERS
53 //-----

54 bool Record::operator< (const Record& r2)const{
55     return (key < r2.getKey());
56 }

57 bool Record::operator> (const Record& r2)const{
58     return (key > r2.getKey());
59 }

60 bool Record::operator==(const Record& r2)const{
61     return(key == r2.getKey());
62 }
printf "\\n\\n"

```

cat -b Item.h

```

1 //-----
2 //
3 // Item.h
4 // p4
5 //
6 //-----

7 #ifndef __p4__item__

```

```

8      #define __p4__item__
9      #include <stdio.h>
10     #include <iostream>
11     #include <cstdlib>
12     #include <fstream>
13
14     using namespace std;
15
16     class Item{
17     public:
18         Item();
19         Item(int stockNum, string description, int qty, double cost);
20         int getStockNum() const;
21         string getDescription() const;
22         int getCount() const;
23         double getCost() const;
24     private:
25         int stockNum;
26         char description[8];
27         int count;
28         double cost;
29         void setStockNum(int n);
30         void setDescription(string str);
31         void setCount(int n);
32         void setCost(double n);
33     };
34
35     #endif /* defined(__p4__item__) */
36
37     printf "\\n
38
39     cat -b Item.cpp

```

```

1      //-----
2      //
3      //  Item.cpp
4      //  p4
5      //
6      //-----
7
8      #include "Item.h"

```

```

8      Item::Item(){
9          //-----
10         // Default Item Constructor
11         //-----
12
13         setStockNum(0);
14         setDescription("");
15         setCount(0);
16         setCost(0);
17     }
18
19     Item::Item(int stock, string desc, int qty, double c){
20         //-----
21         //Preconditions: 3 integers and a string are passed to the constructor
22         //                  from the calling code
23         //Postconditions: an Item object is instantiated with the passed values
24         //
25         //Variables used: stock - integer representing an item's stock number
26         //                  desc - description of an item, in a string
27         //                  qty - integer value with the current quantity
28         //                  nextItem - next item in the stock list
29         //-----
30         setStockNum(stock);
31         setDescription(desc);
32         setCount(qty);
33         setCost(c);
34     }
35
36     //-----
37     //BEGIN GETTERS AND SETTERS
38     //-----
39
40     int Item::getStockNum()const{
41         return stockNum;
42     }
43
44     string Item::getDescription()const{
45         return description;
46     }
47
48     int Item::getCount()const{
49         return count;
50     }
51
52     double Item::getCost()const{
53         return cost;
54     }
55
56     void Item::setStockNum(int stock){
57         stockNum = stock;
58     }
59
60     void Item::setDescription(string desc){
61         description = desc;
62     }
63
64     void Item::setCount(int qty){
65         count = qty;
66     }
67
68     void Item::setCost(double c){
69         cost = c;
70     }
71
72     void Item::setNextItem(Item* nextItem){
73         nextItem = nextItem;
74     }
75
76     void Item::setNextItem(int stock, string desc, int qty, double c){
77         Item* newItem = new Item(stock, desc, qty, c);
78         nextItem = newItem;
79     }

```

```

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>

```

```

10     class CreateIndex{
11     public:

12         CreateIndex();
13         void run();
14         void printInventory(list <Record> myList, fstream& myFile);
15         int getRecords();
16         void createBinaryFile(int numRecords);
17     private:
18         int records;
19     };
20 #endif
printf "\\n

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

 7  #include "Item.h"
 8  #include "Record.h"
 9  #include "CreateIndex.h"

10  CreateIndex::CreateIndex(){
11      //-----
12      //  DEFAULT CONSTRUCTOR
13      //-----
14  }

15  void CreateIndex::run() {

16      //-----
17      //Preconditions: The calling code has called this method
18      //
19      //Postconditions: An index file containing
20      //
21      //Variables used: infile1: fstream object that accesses prog5.idx
22                      infile2: fstream object to the data file prog5.dat
23                      numRecords: number of records in the data file
24                      keyArr[]: an integer array used to store keys
25                      item: used to reference a Record object
26                      bsearchResult: integer containing the result

```

```

27         //             of the binarySearchMethod
28         //             stockNum: int to hold the Stock value
29         //             description: string description of the current record
30         //             count: count value for the current record
31         //             cost: double value of the cost of the current record
32         //
33         //-----
34
35         list <Record> myList;
36         Record record;
37         Item item;
38         int lines = 0;
39
40         //open fstream of input file
41
42
43         fstream infile1("../instr/prog5.dat", ios::in);
44         if (!infile1.is_open()) {
45             cout<<"could not open prog5.dat"<<endl;
46         }else{
47             int numRecords;
48             infile1>>numRecords;
49             records = numRecords;
50             infile1.close();
51             list <int> myList1;
52
53             fstream infile2("../instr/prog5.dat", ios::in);
54
55
56             createBinaryFile(numRecords);
57
58
59             fstream myFile("prog5bin.dat", ios::in|ios::out|ios::binary);
60
61
62             int stockNum;
63             string description;
64             int count;
65             double cost;
66             int n=0;
67
68
69             while (infile2 >> stockNum >> description >> count >> cost) {
70
71

```



```

72
73     item = Item(stockNum, description, count, cost);
74     //Creates Item objects and writes the information contained within the object to the output file
75     record = Record(stockNum, lines);
76
77     //following line moves the filepointer to the item's location in bytes on the list
78     myFile.seekg(lines*sizeof(Item));
79
80     //following line writes the information to the output file
81     myFile.write(reinterpret_cast<const char *>(&item), sizeof(Item));
82
83
84
85     myList1.push_back(stockNum);
86
87     myList.push_back(record);
88
89     n++;
90     lines++;
91 }
92
93 infile2.close();//close input file
94
95
96 //following block of code creates the output file
97 ofstream openFile;
98 openFile.open("prog5.idx", ios::out);
99 openFile.close();
100 //end creation of output file
101
102 fstream indexFile("prog5.idx", ios::out);
103 myList.sort();
104 printInventory(myList, indexFile);
105
106 cout<<endl;
107
108 }
109
110 }
111
112 void CreateIndex::printInventory(list<Record> myList, fstream& myFile){
113     //-----
114     //Preconditions: a reference to an output stream and a reference
115     //                to List were passed by the calling code

```

```

111         //Postconditions: the contents of the list are sent to
112         //                 the output stream
113         //
114         //Variables used: &myfile: reference to an output stream
115         //                 item: reference to an Item object
116         //-----
117
118         for(std::list<Record>::iterator it = myList.begin(); it!= myList.end(); ++it)
119         {
120             Record item = *it;
121             if(item.getOffset()>0){
122                 myFile <<left << setw(6)<< item.getKey()<<setw(2)<< item.getOffset()<<endl;
123             }
124         }
125     }
126
127     int CreateIndex::getRecords(){
128         return records;
129     }
130
131     void CreateIndex::createBinaryFile(int numRecords){
132         //-----
133         //Preconditions: number of records in the data file is passed by
134         //                 the calling code
135         //Postconditions: a binary file is created that has the same
136         //                 information as the data file
137         //
138         //Variables used: openFile: ofstream used to create the binary file
139         //                 item: reference to an Item object
140         //-----
141
142         ofstream openFile;
143         openFile.open("prog5bin.dat", ios::out|ios::binary);
144         openFile.close();
145
146         Item item = Item();
147         fstream myFile("prog5bin.dat", ios::in|ios::binary);
148
149         int i = 0;
150         for(i = 0; i<numRecords+1;i++){
151             myFile.write(reinterpret_cast< const char * >(&item), sizeof(Item));
152         }
153     }

```

```
printf "\\n\\n
```

```
cat -b SearchIndex.h
```

```
1 //-----
2 //
3 // SearchIndex.h
4 // p5
5 //
6 //-----
7
8 #ifndef __p5__SearchIndex__
9 #define __p5__SearchIndex__
10
11 #include "Item.h"
12 #include "Record.h"
13 #include <sstream>
14 #include <list>
15
16 using namespace std;
17
18 class SearchIndex{
19 public:
20     SearchIndex();
21     void run(string searchKey, string fileName, int records);
22     int binarySearch(int keyArr[], int numRecords, int key);
23     void getRecord(int offset);
24     void outputLine(const Item &record);
25     int convert(const string& str);
26 };
27
28 #endif /* defined(__p5__SearchIndex__) */
```

```
printf "\\n
```

```
cat -b SearchIndex.cpp
```

```
1 //-----
2 //
3 // SearchIndex.cpp
4 // p5
5 //
6 //-----
7
8 #include "Record.h"
```

```

8      #include "Item.h"
9      #include "SearchIndex.h"

10     SearchIndex::SearchIndex(){
11         //-----
12         // DEFAULT CONSTRUCTOR
13         //-----
14     }

15     void SearchIndex::run(string searchKey, string fileName, int records){
16         //-----
17         //Preconditions: A key string values, a string containing a filename
18         //                  of an index file, and the total number of records
19         //
20         //Postconditions: The integer value of the search key is returned
21         //                  if found, else 0 is returned to the calling code
22         //
23         //Variables used: myFile: fstream object that accesses prog5.idx
24         //                  infile: fstream object to the data file prog5.dat
25         //                  sKey: integer value of the search key
26         //                  offset: integer containing the RRN (offset)
27         //                  keyArr[]: an integer array used to store keys
28         //                  item: used to reference a Record object
29         //                  bsearchResult: integer containing the result
30         //                  of the binarySearchMethod
31         //
32         //-----
33
34         fstream myFile(fileName.c_str(), ios::in);
35         fstream infile("../instr/prog5.dat", ios::in);
36
37         int sKey;
38         sKey = convert(searchKey);
39         int n=0;
40         int key, offset;
41
42         int keyArr[records];
43
44         int i = 0;
45
46         Record item;
47
48         while (myFile >> key >> offset) {
49             while(i!=offset){
50                 i++;

```

```

51         }
52
53         //Creates Item objects and writes the information contained within the object to the output file
54         keyArr[i] = key;
55         i=0;
56         n++;
57
58     }
59
60     int bSearchResult = binarySearch(keyArr, records, sKey);
61
62     if (bSearchResult > 0) {
63         getRecord(bSearchResult);
64
65     } else {
66         cout << "Key not found, please try again." << endl;
67     }
68
69
70
71 }
72
73 int SearchIndex::binarySearch(int keyArr[], int numRecords, int key){
74     //-----
75     //Preconditions: An array of key values, a total number of records
76     //                and a key to be search for is passed to the method
77     //
78     //Postconditions: The integer value of the search key is returned
79     //                if found, else 0 is returned to the calling code
80     //
81     //Variables used: mid: middle index within the array to be compared
82     //                lower: leftmost index of the array to be compared
83     //                upper: rightmost index of the array to be compared
84     //-----
85     int mid, lower = 0;
86     int upper = numRecords;
87     while( lower <= upper )
88     {
89         mid = ( lower + upper )/2;
90         if( key > keyArr[mid] )
91             lower = mid+1;
92         else if(key < keyArr[mid])
93             upper = mid-1;
94         else

```

```

95         return mid;
96     }
97     return 0;
98 }

99 void SearchIndex::getRecord(int offset){
100     //-----
101     //Preconditions: An integer is passed to the method
102     //
103     //Postconditions: The record referenced by the passed integer is
104     //                  printed to the screen
105     //
106     //Variables used: inputFile: reference to an input file stream
107     //                  &item reference to an Item object
108     //-----
109
110     Item item;
111
112     fstream inputFile("prog5bin.dat", ios::in|ios::binary);
113     inputFile.seekg(offset*sizeof(item));
114
115     inputFile.read(reinterpret_cast<char * >(&item), sizeof(Item));
116     outputLine(item);
117     inputFile.close();
118 }

119 void SearchIndex::outputLine(const Item &record){
120     //-----
121     //Preconditions: a reference to an output stream and a reference
122     //                  to an Item object were passed by the calling code
123     //Postconditions: the contents of the record object are sent to
124     //                  the output stream
125     //
126     //Variables used: &output: reference to an output stream
127     //                  &record reference to an Item object
128     //-----
129     cout << right << setw(5)<< record.getStockNum()<<right <<setw(8) << record.getDescription() << right <<
setw(3)<< record.getCount()<< right<<setw(6)<<record.getCost()<<endl;
130
131     cout.clear();
132
133
134 }

135 int SearchIndex::convert(const string& str) {

```

```
136         stringstream ss(str);
137         int n;
138         ss >> n;
139         return n;
140     }
```

```
printf "\\n\\n
```

```
cat -b p5.cpp
```

```
1  /*
2  PROGRAM NAME: Program 5: Indexed Files
3
4  PROGRAMMER:   James Francis
5
6  CLASS:       CSC 331.001, Fall 2014
7
8  INSTRUCTOR:  Dr. Robert Strader
9
10 DATE STARTED: October 21, 2014
11
12 DUE DATE:    October 28, 2014
13
14 PROGRAM PURPOSE:
15
16 1) Create a binary file from prog5.dat
17 2) Read in records from prog5.dat to Record objects to a list
18 3) Write the Item objects to the previously created binary file
19 4) Print to console the item record in the file going from one
20 item to the next based on their nextItem variable
21
22
23 VARIABLE DICTIONARY:
24
25 indexer: CreateIndex object that will create an index of records within the data file
26 searcher: SearchIndex object that will perform search operations
27 key: String holds the value of the key to be searched for
28 fileName: String holds the value of the index file
29
30 ADTs: std::list
31
32 FILES USED: prog5.dat
33
34
35 SAMPLE INPUTS:
```

```
36
37 search 12382 prog5.idx
38
39
40 SAMPLE OUTPUTS:
41
42 12382 Item09 62 41.37
43
44
45 -----*/

46 #include "Record.h"
47 #include "Item.h"
48 #include "CreateIndex.h"
49 #include "SearchIndex.h"

50 int main(int argc, const char * argv[]){
51
52     CreateIndex indexer = CreateIndex();
53     indexer.run();
54
55     SearchIndex searcher = SearchIndex();
56     string key = argv[1];
57     string fileName = argv[2];
58
59     searcher.run(key, fileName, indexer.getRecords());
60
61     return 0;
62 }
```