```
 1      /*
 2       PROGRAM NAME: Card.h
 3
 4       PROGRAMMER:   James Francis
 5
 6       CLASS:        CSC 331.001, Fall 2014
 7
 8       INSTRUCTOR:   Dr. Robert Strader
 9
10       DATE STARTED: November 1, 2014
11
12       DUE DATE:     November 6, 2014
13
14       PROGRAM PURPOSE:
15       Declaration of the Card Class
16
17       VARIABLE DICTIONARY:
18
19
20       ADTs: none
21
22       FILES USED:
23
24
25       SAMPLE INPUTS:
26
27       SAMPLE OUTPUTS:

28
29
30       ------------------------------------------------------------------------*/

31      #ifndef __p6__Card__
32      #define __p6__Card__

33      #include <stdio.h>
34      #include <iostream>
35      #include <iomanip>
36      #include <string>

37      using namespace std;

38      class Card{

39      public:
```

```
    40          Card();
    41          Card(int, string);
    42          int getKey();
    43          string getEntry();
    44          void display();
    45
    46      private:
    47          void setKey(int);
    48          void setEntry(string);
    49          int Key;
    50          string Entry;
    51
    52      };

    53      #endif /* defined(__p6__Card__) */
printf \\n

cat -b Card.cpp
     1      /*
     2       PROGRAM NAME: Card.cpp
     3
     4       PROGRAMMER:   James Francis
     5
     6       CLASS:        CSC 331.001, Fall 2014
     7
     8       INSTRUCTOR:   Dr. Robert Strader
     9
    10       DATE STARTED: November 1, 2014
    11
    12       DUE DATE:     November 6, 2014
    13
    14       PROGRAM PURPOSE:
    15       This file contains the class definition for the Card Class.
    16
    17       VARIABLE DICTIONARY:
    18
    19
    20       ADTs: none
    21
    22       FILES USED: none
    23
    24
    25       SAMPLE INPUTS(from prog6.dat):

    26
```

```
27       SAMPLE OUTPUTS:(to console)

28
29
30       ---------------------------------------------------------------------*/

31      #include "Card.h"
32      Card::Card(){
33          //--------------------
34          //DEFAULT CONSTRUCTOR
35          //--------------------
36          setKey(0);
37          setEntry("");
38      }

39      Card::Card(int n, string str){
40          //------------------------
41          //INITIALIZING CONSTRUCTOR
42          //------------------------
43          setKey(n);
44          setEntry(str);
45      }


46      void Card::display(){
47          //-------------------------
48          // DISPLAYS THE CARD'S FACT
49          //-------------------------

50
51          printf("%s ", Entry.c_str());


52      }

53      //--------------------
54      //   GETTERS
55      //--------------------
56      int Card::getKey(){
57          return Key;
58      }
59      string Card::getEntry(){
60          return Entry;
61      }
```

```
    62      //--------------------
    63      //   SETTERS
    64      //--------------------
    65      void Card::setKey(int n){
    66          Key = n;
    67      }
    68      void Card::setEntry(string str){
    69          Entry = str.c_str();
    70      }
printf \\n\\n


cat -b HyperCardStack.h
     1      /*
     2       PROGRAM NAME: HyperCardStack.h
     3
     4       PROGRAMMER:   James Francis
     5
     6       CLASS:        CSC 331.001, Fall 2014
     7
     8       INSTRUCTOR:   Dr. Robert Strader
     9
    10       DATE STARTED: November 1, 2014
    11
    12       DUE DATE:     November 6, 2014
    13
    14       PROGRAM PURPOSE:
    15       Declaration for the HyperCardStack class. Also includes a declaration
    16       for a Node struct to be used by calling code. As a way to move through
    17       the list.
    18
    19       VARIABLE DICTIONARY:
    20
    21       ADTs: none
    22
    23       FILES USED:

    24
    25
    26       ---------------------------------------------------------------------*/

    27      #ifndef __p6__HyperCardStack__
    28      #define __p6__HyperCardStack__

    29      #include <stdio.h>
```

```
    30      #include <iomanip>
    31      #include <fstream>
    32      #include <iostream>
    33      #include <sstream>
    34      #include "Card.h"

    35      using namespace std;

    36      struct Node{
    37          Card data;
    38          Node* next;
    39      };

    40      class HyperCardStack{
    41      public:
    42          HyperCardStack();
    43          void insert(int, string);
    44          void remove(int);
    45          void traverse();
    46          void forward();
    47          void home();
    48          void print();
    49
    50      private:
    51          int count;
    52          void emptyInsert(Card);
    53          void stdInsert(Card);
    54          void printPointers();
    55          Node *Current;
    56          Node *Tail;
    57      };




    58      #endif /* defined(__p6__HyperCardStack__) */
printf \\n

cat -b HyperCardStack.cpp
     1      /*
     2       PROGRAM NAME: HyperCardStack.cpp
     3
     4       PROGRAMMER:   James Francis
     5
     6       CLASS:        CSC 331.001, Fall 2014
```

```
 7
 8       INSTRUCTOR:   Dr. Robert Strader
 9
10       DATE STARTED: November 1, 2014
11
12       DUE DATE:     November 6, 2014
13
14       PROGRAM PURPOSE:
15       This file contains the class definition for the HyperCardStack Class.
16
17       VARIABLE DICTIONARY:
18
19
20       ADTs: HyperCardStack
21
22       FILES USED: none
23
24
25       ---------------------------------------------------------------------*/

26      #include "HyperCardStack.h"

27      HyperCardStack::HyperCardStack(){
28          //---------------------------
29          // Default Constructor
30          //---------------------------
31          Current = new Node;
32          Tail = new Node;
33          count = 0;
34      }

35      void HyperCardStack::insert(int n, string str){
36          //---------------------------------------------------------------------
37          //Preconditions: Calling code calls the HyperCardStack insert function
38          //
39          //Postconditions: A new Card object was added as part of a Node within
40          //              the HyperCardStack
41          //
42          //Variables used:
43          //              count: integer that stores the current count of nodes in the
44          //                 HyperCardStack
45          //              newCard: object containing the new Card to be added
46          //---------------------------------------------------------------------
47
48
```

```
49          Card newCard = Card(n, str);
50          cout<<"Inserting: "<<newCard.getKey()<<newCard.getEntry();
51          if (count == 0) {
52              emptyInsert(newCard);
53              count++;
54          }else{
55              stdInsert(newCard);
56
57          count++;
58          }
59
60          cout<<endl;
61      }

62      void HyperCardStack::remove(int n){
63          //-----------------------------------------------------------------------
64          //Preconditions: Calling code calls the HyperCardStack remove function
65          //
66          //Postconditions: The node that contains the integer passed by the calling
67          //                  code has been removed from the HypercardStack
68          //
69          //Variables used:
70          //              k: integer that stores the current count of nodes in the
71          //                 HyperCardStack
72          //              ptr: Node that maintains a reference to the Current Node's
73          //                 location, prior to printing all the requested data
74          //                Tail: Node that points to the last Card object
75          //                Current: Node that points to the current Card object
76          //-----------------------------------------------------------------------
77
78          int k = count;
79          Node* ptr;
80          ptr = Current;
81
82          if (count ==1) {
83              Current = NULL;
84              Tail = NULL;
85          }
86
87          while (ptr->next->data.getKey() != n && k>0) {
88              ptr = ptr->next;
89              k--;
90          }
91          if (ptr->next->data.getKey() !=n) {
92              cout<< "Key not found in the list.";
```

```
 93          }else
 94
 95          cout<<"Removing: "<<ptr->next->data.getKey()<<ptr->next->data.getEntry();
 96          cout <<endl;
 97          ptr->next = ptr->next->next;
 98          count--;
 99          Current = ptr;

100      }

101      void HyperCardStack::traverse(){
102          //-----------------------------------------------------------------------
103          //Preconditions: Calling code calls the HyperCardStack traverse function
104          //
105          //Postconditions: All facts contained in the Card objects, referenced by nodes:
106          // beginning with the Current Node and ending with the node before Current
107          //
108          //Variables used: ptr: Node that maintains a reference to the Current Node's
109          //                 location, prior to printing all the requested data
110          //                Current: Node that points to the current Card object
111          //-----------------------------------------------------------------------
112          Node *ptr = new Node;
113          ptr = Current;
114          cout<<"Traversing: ";

115          do {
116                  Current -> data.display();
117                  Current = Current -> next;
118
119              }while(Current != ptr);
120          cout.clear();
121          cout<<endl;
122          Current = ptr;
123      }

124      void HyperCardStack::forward(){
125          //-----------------------------------------------------------------------
126          //Preconditions: The calling code has invoked the forward method
127          //
128          //Postconditions: The current pointer now references the node at
129          //                 Current->next
130          //
131          //Variables used:
132          //                  Current: Node that points to the current Card object
133          //-----------------------------------------------------------------------
```

```
134         cout<<"Moving the current pointer forward.";
135         cout<<endl;
136         Current = Current->next;
137     }
138     void HyperCardStack::home(){
139         //-------------------------------------------------------------------------
140         //Preconditions: The calling code has invoked the home method
141         //
142         //Postconditions: The current pointer now references the node at
143         //                 Tail->next
144         //
145         //Variables used:
146         //                  Tail: Node that points to the last Card object
147         //                  Current: Node that points to the current Card object
148         //-------------------------------------------------------------------------
149         cout<<"Moving the current pointer home.";
150         cout<<endl;
151         Current = Tail->next;
152     }
153     void HyperCardStack::emptyInsert(Card newCard){
154         //-------------------------------------------------------------------------
155         //Preconditions: a new Card object has been instantiated by the
156         //               calling code
157         //
158         //Postconditions: The passed Card object has been added as the
159         //                Tail element in the HyperCardStack
160         //
161         //Variables used:   newCard: Reference to a Card Object
162         //                  temp: Node containing a reference to newCard
163         //                  Tail: Node that points to the last Card object
164         //                  Current: Node that points to the current Card object
165         //-------------------------------------------------------------------------
166
167         Node *temp = new Node;
168         temp->data = newCard;
169         temp->next = Tail;
170         Tail = temp;
171         Tail->next = Tail;
172
173         Current = Tail;
174         Current->next = Tail;
```

```
175     }

176     void HyperCardStack::stdInsert(Card newCard){
177         //-----------------------------------------------------------------------
178         //Preconditions: a new Card object has been instantiated by the
179         //               calling code
180         //
181         //Postconditions: The passed Card object has been added as the
182         //                Tail element in the HyperCardStack
183         //
184         //Variables used:   newCard: Reference to a Card Object
185         //                  temp: Node containing a reference to newCard
186         //                  Tail: Node that points to the last Card object
187         //                  Current: Node that points to the current Card object
188         //-----------------------------------------------------------------------
189
190         if(Current == Tail){
191         Node  *temp = new Node;//new node ptr
192
193         temp->data = newCard;// newCard object is referenced to by temp node pointer
194         temp->next = Tail->next;// Current
195         Current->next = temp;
196         Current = temp;
197         Tail = Current;

198
199     }
200         else {
201             Node  *temp = new Node;//new node ptr

202             temp->data = newCard;// newCard object is referenced to by temp node pointer
203             temp->next = Current->next;// Current
204             Current->next = temp;


205         }
206     }

207     void HyperCardStack::print(){
208         //-----------------------------------------------------------------------
209         //Preconditions: Calling code calls this Node's object's print method
210         //
211         //Postconditions: The requested Node's object's fact is displayed
212         //
```

```
   213          //Variables used:
   214          //                Current: Node that points to the current Card object
   215          //------------------------------------------------------------------------
   216          cout<<"Printing: ";
   217          Current->data.display();
   218          cout<<endl;

   219      }

printf \\n\\n


cat -b prog6.cpp
     1      /*
     2       PROGRAM NAME: Program 6: Linked Lists
     3
     4       PROGRAMMER:   James Francis
     5
     6       CLASS:        CSC 331.001, Fall 2014
     7
     8       INSTRUCTOR:   Dr. Robert Strader
     9
    10       DATE STARTED: November 1, 2014
    11
    12       DUE DATE:     November 6, 2014
    13
    14       PROGRAM PURPOSE:
    15       This program is used to implement to Linked List class, and Card class, for use in a Hyperstack-like Structure.
    16       This program will read in data from prog6.dat and perform operations on the hyperstack based upon input.
    17
    18       VARIABLE DICTIONARY:
    19
    20       stack: Reference to a HyperCardStack Object
    21       command: charcater containing the command to be performed
    22       key: key value to be inserted or deleted
    23       entry: string containing a fact
    24       line: string containin a line of input from the input file
    25
    26       ADTs: HyperCardStack
    27
    28
    29       FILES USED: prog6.dat
    30
    31
    32       SAMPLE INPUTS(from prog6.dat):
```

```
i 27 Mary had a little lamb
i 15 Today is a good day
i 35 Now is the time!
i 9 This lab is easy and fun
p
d 35
t
i 37 Better Now.
f
p
h
p
d 27
d 15
d 37
d 9
i 44 This should be it!
t
p

SAMPLE OUTPUTS:(to console)

Inserting: 27

Inserting: 15

Inserting: 35

Inserting: 9

Printing:  This lab is easy and fun

Removing: 35

Traversing:  Today is a good day   This lab is easy and fun   Mary had a little lamb

Inserting: 37

Moving the current pointer forward.

Printing:  Better Now.

Moving the current pointer home.
```

```
 78       Printing:  Mary had a little lamb
 79
 80       Removing: 27
 81
 82       Removing: 15
 83
 84       Removing: 37
 85
 86       Removing: 9
 87
 88       Inserting: 44
 89
 90       Traversing:  This should be it!
 91
 92       Printing:  This should be it!

 93
 94       ----------------------------------------------------------------------*/

 95      #include "Card.h"
 96      #include "HyperCardStack.h"

 97      void parseInput(HyperCardStack& stack,char c, int key, string entry);


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

 99          HyperCardStack stack = HyperCardStack();

100          fstream infile("../instr/prog6.dat", ios::in);
101          if (!infile.is_open()) {
102              cout<<"File not found."<<endl;
103              return -1;
104          }
105          string line;
106          char command;
107          int key;
108          string entry;
109
110
111          while (!infile.eof()) {
112              command='\0';
113              key='\0';
114              entry=" ";
115
```

```
116             getline(infile, line);
117             stringstream linestream(line);
118             linestream>>command;
119             linestream>>key;
120             string str;
121             while (linestream>>str) {
122                 str+=" ";
123                 entry+=str.c_str();
124             }
125 
126             parseInput(stack, command, key, entry);
127         }

128         infile.close();


129            return 0;
130     }

131     void parseInput(HyperCardStack& stack,char c, int key, string entry){
132         //-----------------------------------------------------------
133         //Preconditions: a reference to a HyperCardStack, a character,
134         //       an integer and a string are passed by the calling code.
135         //
136         //Postconditions: An insert, delete, traverse, home, forward or
137         //             print are performed on the passed HyperCardStack.
138         //
139         //Variables used:   stack: Reference to a HyperCardStack Object
140         //                  c: charcater containing the command to be
141         //                      performed
142         //                  key: key value to be inserted or deleted
143         //                  entry: string to be inserted
144         //-----------------------------------------------------------

145 
146         switch (c) {
147             case 'i':
148                 stack.insert(key, entry);
149                 cout<<endl;
150                 break;
151 
152             case 'd':
153                 stack.remove(key);
154                 cout<<endl;
155                 break;
156 
```

```
   157              case 't':
   158                  stack.traverse();
   159                  cout<<endl;
   160                  break;
   161
   162              case 'h':
   163                  stack.home();
   164                  cout<<endl;
   165                  break;
   166
   167              case 'f':
   168                  stack.forward();
   169                  cout<<endl;
   170                  break;
   171
   172              case 'p':
   173                  stack.print();
   174                  cout<<endl;
   175                  break;
   176
   177              default:
   178                  break;
   179          }

   180      }

:
g++ Card.cpp HyperCardStack.cpp prog6.cpp -o prog6
:
prog6
Inserting: 27 Mary had a little lamb

Inserting: 15 Today is a good day

Inserting: 35 Now is the time!

Inserting: 9 This lab is easy and fun

Printing:  This lab is easy and fun

Removing: 35 Now is the time!

Traversing:  Today is a good day   This lab is easy and fun   Mary had a little lamb

Inserting: 37 Better Now.
```

```
Moving the current pointer forward.

Printing:  Better Now.

Moving the current pointer home.

Printing:  Mary had a little lamb

Removing: 27 Mary had a little lamb

Removing: 15 Today is a good day

Removing: 37 Better Now.

Removing: 9 This lab is easy and fun

Inserting: 44 This should be it!

Traversing:  This should be it!

Printing:  This should be it!
```