```cpp
/*
Filename: p7.cpp
Author(s): Zachary Rea and Parker Ross
Date: March 7 2023
Description:
*/
#include "P7.h"
#include <iostream>

using namespace std;

//*****************************************************************************
//Constructors and Destructors
//*****************************************************************************
//Constructor
//Written by Zach
intList::intList (int listCapacity) {
    this->listCapacity = listCapacity;
    a = new int[listCapacity];
    listSize = 0;
}

//*****************************************************************************
//Destructor
//Written by Zach
intList::~intList() {
    delete[] a;
}

//*****************************************************************************
//Public Functions
//*****************************************************************************
//Function to insert a value at the start of the list
//Written by Parker modified by Zach
bool intList::insert(int key) {
    bool rc = listCapacity > listSize;
    if (rc) {
        for (int i = listSize; i > 0; i--) {
            a[i] = a[i-1];
        }
        a[0] = key;
        listSize++;
    }
    return rc;
}

//*****************************************************************************
//Fucntion to add an int to the end of the list
//Written by Parker modified by Zach
bool intList::add(int key) {
    bool rc;
    if (listSize < listCapacity) {
        a[listSize] = key;
        listSize++;
        rc = true;
    }
    return rc;
}

//*****************************************************************************
//Function to insert a value at a given index of the list
//Written by Parker modified by Zach
bool intList::insertAt(int index, int key) {
    bool rc = listCapacity > listSize && index < listSize && index >= 0;
    if (rc) {
        for (int i = listSize; i > index; i--) {
            a [i] = a[i-1];
        }
        a[index] = key;
```

```
70          listSize++;
71      }
72      return rc;
73  }
74
75  //*************************************************************************
76  //Function to delete a value at a given index of the list
77  //Written by Parker modified by Zach
78  bool intList::deleteAt(int index, int &key) {
79      bool rc = listSize > index && index >= 0;
80          if (rc) {
81              key = a[index];
82              for (int i = index; i < listSize; i++) {
83                  a[i] = a[i+1];
84              }
85          listSize--;
86      }
87      return rc;
88  }
89
90  //*************************************************************************
91  //Function to set the size to 0
92  //Written and modified by Zach
93  void intList::clear() {
94      listSize = 0;
95  }
96
97  //*************************************************************************
98  //Function to print n of the list
99  //Written and modified by Zach
100 void intList::printIt(int n) const{
101     cout << "printIt with list size: " << listSize << " capacity = ";
102     cout << listCapacity << "\n";
103     if (n < listSize) {
104         for (int i = 0; i < n; i++) {
105             cout << "At pos " << i << " there is " << a[i] << "\n";
106         }
107         cout << "At pos " << listSize - 1 << " there is " << a[listSize - 1];
108         cout << "\n";
109     } else {
110         for (int i = 0; i < listSize; i++) {
111             cout << "At pos " << i << " there is " << a[i] << "\n";
112         }
113     }
114 }
115
116 //*************************************************************************
117 //Function to return the index of a given value
118 //Written and modified by Zach
119 int intList::getIndex(int key) const{
120     int rc = -1;
121     for (int i = 0;i < listSize - 1;i++) {
122         if (a[i] == key) {
123             rc = i;
124             break;
125         }
126     }
127     return rc;
128 }
129
130 //*************************************************************************
131 //Function to read the value at a given index
132 //Written by Parker modified by Zach
133 int intList::readAt(int index, int &key) const{
134     int rc = -1;
135     if (index < listSize && index >= 0) {
136         rc = index;
137         key = a[index];
138     }
```

```
139       return rc;
140   }
141
142   //*************************************************************************
143   //Function to return the current size of the list
144   //Written and modiefied by Zach
145   int intList::capacity() const{
146       return listCapacity;
147   }
148
149   //*************************************************************************
150   //Fucntion to sort the array via bubble method
151   //Written by Zach
152   void intList::bubbleSort() {
153       if (listSize > 0) {
154           for (int i = listSize; i > 1; i--) {
155               for (int j = 0; j < i-1; j++) {
156                   if (a[j] > a[j+1]) {
157                       swap(&a[j],&a[j+1]);
158                   }
159               }
160           }
161       }
162   }
163
164   //*************************************************************************
165   //Function to sort the array via selection method
166   //Written by Parker
167   void intList::selectionSort() {
168       for (int i = 0, j, least; i < listSize-1; i++){
169           for (j = i+1, least = i; j < listSize; j++)
170               if (a[j] < a[least])
171                   least = j;
172                   swap(&a[least],&a[i]);
173       }
174   }
175   //*************************************************************************
176   //Function to sort the array via the insertion method
177   //Written by Parker
178   void intList::insertionSort() {
179       for (int i = 1, j; i < listSize; i++){
180           int tmp = a[i];
181           for (j = i; j > 0 && tmp < a[j-1]; j--)
182               a[j] = a[j-1];
183               a[j] = tmp;
184       }
185   }
186
187   //*************************************************************************
188   //Function to tell if the array is currently sorted
189   //Written by Zach
190   bool intList::isSorted() const{
191       bool rc;
192       for (int i = 0;i < listSize - 1; i++) {
193           if (a[i] <= a[i + 1]) {
194               rc = true;
195           } else {
196               rc = false;
197               cout << "Index " << i << " is greater than index " << i + 1 << " ";
198               break;
199           }
200       }
201       return rc;
202   }
203
204   //*************************************************************************
205   //Function to swap two values
206   //Written by Zach
207   void intList::swap(int *i, int *j) const{
```

```
208         int s = *i;
209         *i = *j;
210         *j = s;
211     }
```