```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;
        listSize++;
    }
    return rc;
}

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

//****************************************************************************
//Function to set the size to 0
//Written and modified by Zach
void intList::clear() {
    listSize = 0;
}

//****************************************************************************
//Function to print n of the list
//Written and modified by Zach
void intList::printIt(int n) const{
    cout << "printIt with list size: " << listSize << " capacity = ";
    cout << listCapacity << "\n";
    if (n < listSize) {
        for (int i = 0; i < n; i++) {
            cout << "At pos " << i << " there is " << a[i] << "\n";
        }
        cout << "At pos " << listSize - 1 << " there is " << a[listSize - 1];
        cout << "\n";
    } else {
        for (int i = 0; i < listSize; i++) {
            cout << "At pos " << i << " there is " << a[i] << "\n";
        }
    }
}

//****************************************************************************
//Function to return the index of a given value
//Written and modified by Zach
int intList::getIndex(int key) const{
    int rc = -1;
    for (int i = 0;i < listSize - 1;i++) {
        if (a[i] == key) {
            rc = i;
            break;
        }
    }
    return rc;
}

//****************************************************************************
//Function to read the value at a given index
//Written by Parker modified by Zach
int intList::readAt(int index, int &key) const{
    int rc = -1;
    if (index < listSize && index >= 0) {
        rc = index;
        key = a[index];
    }
    return rc;
}

//***************************************************************************
//Function to return the current size of the list
//Written and modiefied by Zach
int intList::capacity() const{
    return listCapacity;
}

//***************************************************************************
//Fucntion to sort the array via bubble method
//Written by Zach
void intList::bubbleSort() {
    if (listSize > 0) {
        for (int i = listSize; i > 1; i--) {
            for (int j = 0; j < i-1; j++) {
                if (a[j] > a[j+1]) {
                    swap(&a[j],&a[j+1]);
                }
            }
        }
    }
}

//***************************************************************************
//Function to sort the array via selection method
//Written by Parker
void intList::selectionSort() {
    for (int i = 0, j, least; i < listSize-1; i++){
        for (j = i+1, least = i; j < listSize; j++)
            if (a[j] < a[least])
                least = j;
                swap(&a[least],&a[i]);
    }
}
//***************************************************************************
//Function to sort the array via the insertion method
//Written by Parker
void intList::insertionSort() {
    for (int i = 1, j; i < listSize; i++){
        int tmp = a[i];
        for (j = i; j > 0 && tmp < a[j-1]; j--)
            a[j] = a[j-1];
            a[j] = tmp;
    }
}

//***************************************************************************
//Function to tell if the array is currently sorted
//Written by Zach
bool intList::isSorted() const{
    bool rc;
    for (int i = 0;i < listSize - 1; i++) {
        if (a[i] <= a[i + 1]) {
            rc = true;
        } else {
            rc = false;
            cout << "Index " << i << " is greater than index " << i + 1 << " ";
            break;
        }
    }
    return rc;
}

//***************************************************************************
//Function to swap two values
//Written by Zach
void intList::swap(int *i, int *j) const{
    int s = *i;
    *i = *j;
    *j = s;
}
```