Exercises

1. The Unsorted List ADT is to be extended with a Boolean member function, `IsThere`, which takes as a parameter an item of type `ItemType` and determines whether there is an element with this key in the list.
   a. Write the specifications for this function.
   b. Write the prototype for this function.
   c. Write the function definition using an array-based implementation.
   d. Write the function definition using a linked implementation
   e. Describe this function in terms of Big-O.
2. Rather than enhancing the Unsorted List ADTs by adding a member function `IsThere`, you decide to write a client function to do the same task.
   a. Write the specifications for this function.
   b. Write the function definition.
   c. Write a paragraph comparing the client function and the member function (Exercise 1) for the same task.
   d. Describe this function in terms of Big-O.
3. An Unsorted Type ADT is to be extended by the addition of function `SplitLists`, which has the following specifications:

SplitLists(UnsortedType list, ItemType item, UnsortedType& list1, UnsortedType& list2)

Function:	Divides list into two lists according to the key of item.
Preconditions:	list has been initialized and is not empty.
Postconditions:	list1 contains all the items of list whose keys are less than or equal to item's key; list2 contains all the items of list whose keys are greater than item's key.

   a. Implement `SplitLists` as an array-based member function of the Unsorted List ADT.
   b. Implement `SplitLists` as a linked member function of the Unsorted List ADT.
4. Implement SplitLists described in Exercise 3 as a client function.
5. The specifications for the Unsorted List ADT state that the item to be deleted is in the list.

a. Rewrite the specification for `DeleteItem` so that the list is unchanged if the item to be deleted is not in the list.

b. Implement `DeleteItem` as specified in (a) using an array-based implementation.

c. Rewrite the specification for `DeleteItem` so that all copies of the item to be deleted are removed if they exist.

d. Implement `DeleteItem` as specified in (c) using an array-based implementation.

6. Redo Exercise 5(b) and 5(d), using a linked implementation.

7. a. Explain the difference between a sequential and a linked representation of a list.

b. Give an example of a problem for which a sequential list would be the better solution.

c. Give an example of a problem for which a linked list would be the better solution.

8. True or False? If you answer False, correct the statement.

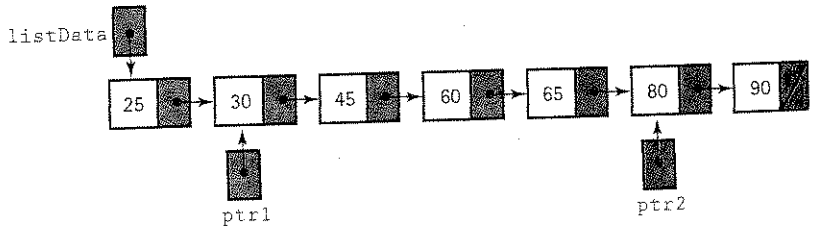
a. An array is a random-access structure.

b. A sequential list is a random-access structure.

c. A linked list is a random-access structure.

d. A sequential list is always stored in a statically allocated structure.

Use the linked list pictured below for Exercises 9 through 12.



9. Give the values of the following expressions:

a. `ptr1->info`

b. `ptr2->next->info`

c. `listData->next->next->info`

10. Are the following expressions true or false?

a. `listdata->next == ptr1`

b. `ptr1->next->info == 60`

c. `ptr2->next == NULL`

d. `listData->info == 25`

11. Decide whether the *syntax* of each of the following statements is valid or invalid. If it is valid, mark it OK; if it is invalid, explain what is wrong.
    a. `listData->next = ptr1->next;`
    b. `listData->next = *(ptr2->next);`
    c. `*listData = ptr2;`
    d. `ptr2 = ptr1->next->info;`
    e. `ptr1->info = ptr2->info;`
    f. `ptr2 = ptr2->next->next;`
12. Write *one* statement to do each of the following:
    a. Make `listData` point to the node containing 45.
    b. Make `ptr2` point to the last node in the list.
    c. Make `listData` point to an empty list.
    d. Set the `info` member of the node containing 45 to 60.

If memory locations are allocated as shown in the second column of the following table, what is printed by the statements in the first column? Fill in the last column in the following table for Exercises 13 through 18. The exercise number is in the first column in comments.

Statements	Memory Allocated	What Is Printed?
`int value;`	value is assigned to location 200	
`value = 500;`		
`char* charPtr;`	charPtr is at location 202	
`char string[10] = "Good luck";`	string[0] is at location 300	
`charPtr = string;`		
`cout << &value;  // Exercise 13`	`& means "the address of"`	
`cout << value;  // Exercise 14`		
`cout << &charPtr;  // Exercise 15`	`& means "the address of"`	
`cout << charPtr;  // Exercise 16`		
`cout << *charPtr;  // Exercise 17`		
`cout << string[2];  // Exercise 18`		

13. An Unsorted List ADT is to be extended by the addition of a member function `Head`, which has the following precondition and postcondition:

    Precondition: list has been initialized and is not empty.

    Postcondition: return value is the last item inserted in the list.

a. Will this addition be easy to implement in the array-based class `UnsortedType`? Explain.

b. Will this addition be easy to implement in the linked class `UnsortedType`. Explain.

14. An Unsorted List ADT is to be extended by the addition of function `Tail`, which has the following precondition and postcondition:

Precondition: list has been initialized and is not empty.

Postcondition: return value is a new list with the last item inserted in the list removed.

a. Will this addition be easy to implement in the array-based class `UnsortedType`? Explain.

b. Will this addition be easy to implement in the linked implementation of class `UnsortedType`? Explain.

15. `DeleteItem` does not maintain the order of insertions because the algorithm swaps the last item into the position of the one being deleted and then decrements `length`. Would there be any advantage to having `DeleteItem` maintain the insertion order? Justify your answer.

16. Give a Big-O estimate of the run time for the functions you wrote in Exercises 5 and 6.

17. a. Change the specifications for the array-based Unsorted List ADT so that `InsertItem` throws an exception if the list is full.

b. Implement the revised specifications in (a).

18. Based on the following declarations, tell whether each statement below is syntactically legal (yes) or illegal (no).

```
int* p;
int* q;
int* r;
int a;
int b;
int c;
```

	Yes/no		Yes/no		Yes/no
a. `p = new int;`	____	f. `r = NULL;`	____	k. `delete r;`	____
b. `q* = new int;`	____	g. `c = *p;`	____	l. `a = new p;`	____
c. `a = new int;`	____	h. `p = *a;`	____	m. `q* = NULL;`	____
d. `p = r;`	____	i. `delete b;`	____	n. `*p = a;`	____
e. `q = b;`	____	j. `q = &c;`	____	o. `c = NULL;`	____

19. The following program has careless errors on several lines. Find and correct the errors and show the output where requested.

```
#include <iostream>
int main ()
{
  int* ptr;
  int* temp;
  int x;

  ptr = new int;
  *ptr = 4;
  *temp = *ptr;
  cout  << ptr  << temp;
  x = 9;
  *temp = x;
  cout  << *ptr  << *temp;
  ptr = new int;
  ptr = 5;
  cout  << *ptr  << *temp;  // output:_____
   return 0;
}
```