1.a	After arriving at a party, you shake hands with each person there						
	assumptions	each handshake takes equal time T					
	total time	n*T					
	order	O(n)					
1.b	Each person in a room shakes hands with everyone else in the room.						
	assumptions	each handshake takes equal time T, ok to repeat handshake					
	person 0 time	n*T					
	total time	n*n*T					
	order	O(n^2)					
1.c	You climb a flight of stairs.						
	assumptions	each step takes equal time T					
	total time	n*T					
	order	O(n)					
1.d	You slide down the banister						
	assumption	travel at constant velocity V for L length					
	total time	L/V					
	order	O(1), it only depends on constant values L and V					
1.e	After entering an elevator, you press a button to choose a fl oor.						
	order	O(1), the operation time is fixed					
1.f	You ride the elevator from the ground fl oor up to the n th fl oor.						
	time for one floor	T					
	time for n flooirs	nT					
	order	O(n)					
1.g	You read a book twice.						
	time to read one book	T					
	time to read 2 books	2T					

	order	O(1)					
2	Describe a way to climb from the bottom of a fl ight of stairs to the top in time that is no better than O(n2).						
	task	time					
	climb stair 1 climb down	2T					
	climb stair 2 climb down	2*2T					
	climb stair n-1 climb down	2*2(n-1)T					
	climb stair n climb down	2*2(n)T					
	Total time	2T(1+2+...n-1 + n)					
		2T(n+1)(n)/2					
	order	O(n^2)					
3.a	Computing the sum of the fi rst n even integers by using a for loop						
	each step	T					
	number of steps	n/2					
	total steps	T/2 *n					
	order	O(n)					
3.b	Displaying all n integers in an array						
	access array	T1					
	display element	T2					
	one step	(T1+T2)					
	n steps	n(T1+T2)					
	order	O(n)					
3.c	Displaying all n integers in a sorted linked chain						
	access sorted list	log2n					
	display element	T2					
	one step	(log2n+T2)					
	n steps	n(nlog2+T2)					

	order	O(nlog2n)					
3.d	Displaying all n names in an array of linked chains						
	access i name	i*T1					
	display element	T2					
	one step	(iT1+T2)					
	n steps	nT2 + (n)(n+1)/2T1					
	order	O(n^2)					
3.e	Displaying one array element						
	fixed time	T					
	order	O(1)					
3.f	Displaying the last integer in a linked chain						
	time to access each node	T					
	n dodes	nT					
	order	O(n)					
3.g	Searching an array of n items for a particular value by using a sequential search						
	each search	T					
	worst case is in last element	nT					
	order	O(n)					
3.h	Searching an array of n items for a particular value by using a binary search						
	each search halves						
	order	O(log2n)					
3.i	Adding an item to a stack of n items						
	worst case have to allocate memory	T1					
	copy n element	nT2					
	add item	T3					
	order	O(n)					

3.j	Adding an item to a bag of n items						
	worst case happens for link chain is full						
	alloc memory	T1					
	copy n element	nT2					
	add item	T3					
	order	O(n)					
4	Suppose that your implementation of a particular algorithm appears in C++ as						
	for (int pass = 1; pass <= n; pass++)						
	{						
	for (int index = 0; index < n; index++)						
	{						
	for (int count = 1; count < 10; count++)						
	{						
	. . .						
	} // end for						
	} // end for						
	} // end for						
	The previous code shows only the repetition in the algorithm, not the computations that occur within the loops.						
	Time for count loop	count-1					
	time for index loop	n(count-1)					
	time for pass loop	nn(count-1)					
	order	O(n^2)					
5	Consider the following C++ function f , which calls the function swap . Assume that swap exists and simply						
	swaps the contents of its two arguments. Do not be concerned with f ’s purpose.						
	void f(int theArray[], int n)						
	{						
		for (int j = 0; j < n; ++j)					n comparisons
		{					

			int i = 0;				
			while (i <= j)				1,2,3,...j
			{				
				if (theArray[i] < theArray[j])			1,2,3,...,j
					swap(theArray[i], theArray[j]);		
				i++;			
			} // end while				
		} // end for					
	} // end f						
	How many comparisons does f perform?						
		sum of 2(1+2+3...n)					
	comparisons	2*n*(n+1)/2					
6	For large arrays, and in the worst case, is a sequential search faster than a binary search? Explain.						
	sequential search takes	n*T					
	binary search takes	log2n*T					
	for n =1024 and T=1						
	sequential search takes	1024					
	binary search takes	10					
	clearly binary search is quicker						