Noah Krill
Data Structures
HW2

```
/*
    1. std::list intersect (const list<object> & L1, const list<object> & L2)
{
        Create list to store intersection.
        Create const iterator and set = to list L1 begin
        Create const iterator2 and set = to list L2 begin
        Create while loop to check if the iterators hit the end
        Check if the value of iterator = the value of iterator two
        If so push the value onto the list that stores the intersections
}
*/

std::list intersect(const std::list<object> & L1, const std::list<object> & L2)
{
std::list<object> L3;
std::list<object>::const_iterator iter = L1.begin();
std::list<object>::const_iterator iter2= L2.begin();
while (iter != L1.end() && iter2 != L2.end())
{
   if (*iter==*iter2)
   {
      L3.pushback(*iter);
         ++iter;
         ++iter2;
   }
   else if (*iter < *iter2)
   {
         ++iter;
   }
   else
   {
         ++iter2;
   }
}
}
```

B.
Well if its not sorted we would have to write a sorting algorithm and that would be Olog(n) then it would be O(n) for the comparing part correct?

2. a-b+c/d -(e–f)*g^h

Symbol	Scanned	Stack	Postfix
1	a		a
2	-	-	a
3	b	-	ab-
4	+	+	ab-
5	c	+	ab-c
6	/	+/	ab-c

7	d	+/	ab-cd
8	-	+-	ab-cd/
9	(	(	ab-cd/+
10	e	(	ab-cd/+e
11	-	(-	ab-cd/+e
12	f	(-	ab-cd/+ef
13	)	-	ab-cd/+ef
14	*	-*	ab-cd/+ef
15	g	-*	ab-cd/+efg
16	^	-* ^	ab-cd/+efg
17	h		a b - c d / + e f g h ^ * -

3.

641-/52^*4–5+

Symbol	Scanned	Stack	Prefix
1	6	6	
2	4	64	
3	1	641	
4	-	6	4-1
5	/		6/(4-1)
6	5	5	6/(4-1)
7	2	52	6/(4-1)
8	^	52^	6/(4-1)
9	*		6/(4-1)*(5^2)
10	4	4	6/(4-1)*(5^2)
11	-		6/(4-1)*(5^2)-4
12	5		6/(4-1)*(5^2)-4
13	+		6/(4-1)*(5^2)-4+5

4.

```
/*
bool remove(stack<Comparable> & st, Comparable obj)
{
   Create a temp stack to hold the values of the items popped off the stack.
   Create a bool return if item is found or not.
   Establish a while loop to loop through the stack until it finds the element or its empty.
   Create if statement to find if the element is on the top of the stack.
   If it is pop of the element and set the found bool to true.
   Else push the value of the top onto the temp stack and pop it off the top of the original stack.
   After the loop is completed push all the elements from the temp stack back onto the original stack.
   Pop of the values of the temp stack.
   Return true if element is removed and false if not.
}
*/
```

Noah Krill
Data Structures
HW2

```
bool remove(stack<Comparable> & st, Comparable obj)
{
   std::stack<Comparable> temp;
   bool found=false;
   while (!stack.empty())
   {
      if (stack.top()==obj)
      {
         stack.pop();
         found=true;
      }
      else
      {
         temp.push(stack.top());
         stack.pop();
      }
   }
   while (!temp.empty())
     {
        stack.push(temp.top());
        temp.pop();
     }
   if(found == true)
   {
      return true;
   }
     return false;
}
```

The reason that STL doesn't have this functionality is because the stack wasn't intended to be used this way and its easy to do this yourself.