Chapter 4

1. Write a function that inverts a single link list.

2. Write a function that concatenate two circular link list A and B to one circular link list C.

3. Rewrite *delete* so that it uses only two pointers, $first$ and $trail$.

```
void delete(listPointer *first, listPointer trail, listPointer x)
{/* delete x from the list, trail is the preceding node
    and *first is the front of the list */
    if (trail)
       trail-->link = x-->link;
    else
      *first = (*first)-->link;
   free(x);
}
```

4. Assume that we have a list of integers as in figure1. Create a function that searches for an integer, num. If num is in the list, the function should return a pointer to the node that contains num. Otherwise it should return $NULL$.

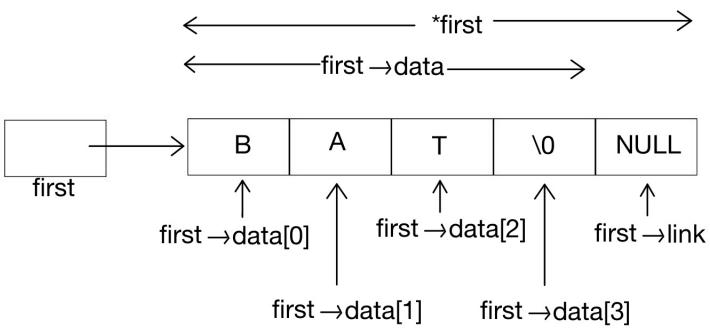

Figure 1:

5. Write a function, *length*, that returns the number of nodes in a list.

6. Write a function, *pread*, that reads in n pairs of coefficients and exponents, ($coef_i$, $expon_i$), $0 \leq i < n$ of a polynomial, x. Assume that $expon_{i+1} > expon_i$, $0 \leq i < n-2$, and that $coef_i \neq 0$, $0 \leq i < n$. Show that this operation can be performed in O(n) time.

7. Let a be a pointer to a polynomial. Write a function, *peval*, to evaluate the polynomial a at point x, where x is some floating point number.

8. Rewrite Exercise 6 using a circular representation for the polynomial.

9. Write a function that deletes a node containing a number, num, from a circularly linked list. Your function should first search for num.