List Operation

1. Write a routine that returns the number of elements of a circular, simply linked list.

2. Write a routine that receives a list and a number X and returns the number of nodes in the list that have the number X. Consider the list to be of the linear, simply linked type.

3. Write a routine that receives a list and a number X and returns the number of nodes in the list that have values greater than the number X. Consider the list to be of the circular, simply linked type.

4. Write a routine that receives a list and a number X and returns the number of nodes in the list that have values smaller than the number X. Consider the list to be of the linear, simply linked type with descriptor.

5. Write a routine that receives a list and returns two lists where one has the odd values and the other the even values of the list. Consider the list being of the following types: linear, circular and with descriptor (which contains a linear list simply linked) 6. Write a routine that receives a list and a number X and returns two lists where one has values smaller than the number X and the other has values greater than X. Consider the list being of the following types: linear, circular and with descriptor (which contains a linear list simply linked) 7. Write a routine that receives a list and a number X and a Y and replaces all occurrences of the number X with the number Y. Return the modified list and the number of times it was replaced. Consider the list being of the following types: linear, circular and with descriptor (which contains a linear list simply linked) 8. Given a list and an element, write a routine that removes from the list all occurrences of the element. Consider the list being of the types: linear, circular and with descriptor (which contains a linear simply linked list) 9. Write a program that allows counting the number of times that there was insertion and removal of elements in a list. Consider the list being of the types: linear and circular simply linked 10. Write a routine that returns the content of the first node of a list. Consider the list being of the types: linear and circular simply linked 11. Write a routine that returns the content of the last node of a list. Consider the list being of the types: linear and circular simply linked 12. Write a routine that returns a pointer to the last node of a list. Consider the list being of the types: linear and circular simply linked 13. Write a routine to insert an element after the n-th element of a list. Consider the list being of the types: linear and circular simply linked 14. Write a routine to insert an element before the n-th element of a list. Consider the list being of the types: linear and circular simply linked

15. Write a routine to remove the n-th element from a list. Consider the list being of the types: linear and circular simply linked

16. Write a routine to combine two ordered lists into a single ordered list. Consider the list being of the types: linear and circular simply linked

17. Write a routine to merge two lists into a single list. Consider the list being of the types: linear and circular simply linked

18. Write a routine to concatenate two lists. Consider the list being of the types: linear and circular simply linked

19. Write a routine to create a copy of a list. Consider the list being of the types: linear and circular simply linked

20. Write a routine that informs if the two lists are identical. Consider the list being of the types: linear and circular simply linked

21. Write a routine that returns the sum of the numbers in a list. Consider the list being of the following types:

linear, circular and with descriptor (which contains a linear simply linked list)

22. Write a routine that returns the product of the numbers in a list. Consider the list being of the following types:

linear, circular and with descriptor (which contains a linear simply linked list)

23. Create a routine that allows you to invert the direction of the references in a linear simply linked list (i.e., the last node becomes the first in the list).

24. Polynomials can be represented by means of lists, whose nodes are records with 3 fields:

coefficient, exponent and reference to the next one. Using this representation, write procedures:

a) given x ∈ R, calculate p(x);

b) add polynomials;

25. Considering the polynomial from the previous exercise, create a routine to determine the value of p(x), where x is a real variable. 26. Write a program that reads two lists (L1 and L2) and an integer N, and prints the list resulting from the insertion of L2 from position N of L1. Consider the list to be of the following types: linear, circular and with descriptor (which contains a simply linked linear list). 27. Write a program to move an element of the List N positions forward. Consider the list being of the following types: linear, circular and with descriptor (which contains a linear list simply linked) 28. Write a program that reads a list L and a value X (belonging to L) and creates another list where the element X was placed in position 0. Consider the list being of the following types: linear, circular and with descriptor (which contains a linear list simply linked) 29. Write a program that reads a list L, a value X (belonging to L) and a value N and creates another list where the element X was placed in position N. Consider the list being of the following types: linear, circular and with descriptor (which contains a linear list simply linked) 30. A usual way to represent a set is by the list of its elements. Assuming this representation, write procedures for the usual set operations: union, intersection and difference. Consider the list being of the

following types: linear and circular simply linked. 31. Let's consider a list of the type Simply Linked Circular List, as defined below:

typedef enum {False,True} boolean;

typedef struct no { int info;

struct no* prox;

} *def_lista;

Build the routines below taking into account the given prototype:

a) def_lista initializes (void) the routine for initializing a list

b) void Inserre(def_lista* List, int number) routine for inserting elements at the beginning of the list

c) boolean Remove(def_lista* List, int* number) routine for removing elements from the beginning of the list

d) void imprime(def_lista List) routine for printing the list

Make a routine that generates a list List1 where the odd elements of List do NOT appear,

respecting the prototype: def_lista cria_lista(def_lista01 List)

only using the routines above. Do not use an auxiliary list to traverse List and create another list.

For example: If List has the elements: {9, 6, 5, 3, 2, 1}, List1 will be {2, 6}

32. There is an ordered list L1 whose elements are words and a list L2 that contains numbers. The problem consists of removing from list L1 the nodes whose positions are marked in list L2. The positions in L2 vary from 1 to N. N may not be a valid position.

For example:

L1 = { structure, data, bcc, students, lab, friends, task, work}

L2 = { 2, 5, 9}

Resulting list = { structure, bcc, students, friends, task, work}

33. Suppose we want to form N Ordered Lists, where N is a constant. Implement the insertion, removal, search and printing routines. Always read two numbers on each input line, the first being the index number of the list in which the second number is to be inserted, removed or searched. When printing, show all N Lists.