LISTS

SPECIFICATION

```
spec LIST[INT]
      genre list, int
      operations
            _=_: int int -> bool
            count_dif_elem: list -> int
            same_lists: list list -> bool
            same_elements: list list -> bool
            rearrange_list: list -> list
endspec
```

IMPLEMENTATION

```
DATATYPES
celltypeint = record
      elementList: int
      next: ^celltypeint
endrecord


class List_int
      private header: ^celltypeint
      public int count_dif_elem(list)
      public bool same_lists(list,list)
      public bool same_elements (list,list)
      public list rearrange_list (list)
endclass
```

```
public list List_int::count_dif_elem(l:list)
      temp,tempaux:position
      temp:=header
      temp.aux:=temp^.next
      l1:list
      l1.insert(l.header.elementList,l1.header)
      e:elementList
      if (temp=null)
            return 0
      while (temp!=null)
            e:= tempaux.elementList
            if (temp.elementList!=e)
                  if (e not in l1)
                        l1.insert(e,l1.last()^.next)
                        tempaux:=tempaux^.next
                  endif
            endif
            if (temp.elementList=e)
                  tempaux:=tempaux.^next
            endif
            if (tempaux^.next=null)
                  temp:=temp^.next
                  tempaux:=temp.^next
            endif
      endwhile
      return sizeof(l1)
endmethod
```

$O(n^3)$

The running time of this method is $O(n^3)$ due to a while loop that is being executing n times, inside the loop there is a call to method last which returns the last element of a list and it's running time is O(n). Also, inside the while loop I'm going through out l1.

```
public bool List_int::same_lists(l1:list,l2:list)
      temp1,temp2:position
      temp1:=l1.header
      temp2:=l2.header
      cont:int
      cont:=0
      for i=0 to i<sizeof(l1)
            if (temp1.elementList=temp2.elementList)
                  cont=cont+1
            endif
            temp1:=temp1^.next
            temp2:=temp2^.next
      endfor
      if (cont=sizeof(l1))
            return true
      endif
      else
            return false
      endelse
endmethod
```

O(n)

The running time of this method is O(n) because it has a for loop that is going to be executed n times.

```
public bool List_int::same_elements(l1:list,l2:list)
      temp1,temp2:position
      temp1:=l1.header
      temp2:=l2.header
      cont:=int
      cont:=0
      while (temp1!=null)
            if (temp1.elementList=temp2.elementList)
                  cont++
                  temp2:=temp2^.next
            endif
            if (temp1!=temp2)
                  temp2:=temp2^.next
            endif
            if (temp2^.next=null)
                  temp1:=temp1^.next
                  temp2:=temp1^.next
            endif
      endwhile
      if (cont=sizeof(l1))
            return true
      endif
      else
            return false
      endelse
endmethod
```

O(n)

The running time of this method is O(n) because it has a for loop that is going to be executed n times.

```
public list List_int::rearrange_list(l:list)
      temp,tempaux,tempaux1:position
      temp:=l.header
      tempaux:=temp^.next
      while (temp!=null)
            if (temp.elementList=tempaux.elementList)
                  l.insert(tempaux.elementList,temp^.next)
                  temp:=temp^.next
                  tempaux1:=tempaux
                  l.delete(tempaux1)
                  tempaux:=tempaux^.next
            endif
            if (temp!=tempaux)
                  tempaux:=tempaux^.next
            endif
            if (tempaux=null)
                  temp:=temp^.next
                  tempaux:=temp^.next
            endif
      endwhile
      return l
endmethod
```

O(n)

The running time of this method is O(n) because it has a while loop that is going to be executed n times.