Some things to add to your linked list module....

Copy `linkedList02.f03` to `linkedList03.f03`

1. Remove `head`, `ptr`, and `tail` from `linkedList03.f03`.
2. Add a new subroutine called `linkedListReal_incrementCurrent` that moves the object's `current` pointer one node forward. Modify `linkedList_getCurrent` to use `Function linkListReal_incrementCurrent` instead of the line

   ```
   if(ASSOCIATED(myLinkedList%current%next))
   myLinkedList%current=>myLinkedList%current%next
   ```

   `Function listLinkReal_incrementCurrent` should take only 1 argument, the `linkedListReal` object.
3. Write a new function called `linkedListReal_hasPrevious` that accepts a single dummy argument that is a `linkedListReal` object. This function should be a logical type and indicate whether or not the `current` pointer in the `linkedListReal` object has a set `prev` pointer. This function will be analogous to the logical function `linkedListReal_hasNext`.
4. Edit `Function linkListReal_incrementCurrent` so that it has a second dummy argument called `incrementBy`. This dummy argument (`incrementBy`) should be an integer and be optional. It will indicate the number of moves to incrment the linkedListReal object's `current` pointer by. It should accept both positive and negative values to move forward and reverse in the list. If it is not sent by the calling program unit, it should default to 1.

Copy `linkedList03.f03` to `linkedList04.f03` -and- Copy `linkedListMod.f03` to `linkedListMod1.f03`

4. Edit `linkedList04.f03` so that the `program` and `end program` lines use the updated program name. Also, edit `linkedListMod.f03` so that the `module` and `end module` lines use the updated module name.
5. Edit `linkedList04` to use `linkedListMod1` instead of `linkedListMod`.
6. Edit `Function linkedListReal_getCurrent` so that it has a new optional dummy argument called `incrementBy`. This argument should default to 0 within function `linkedlistReal_getCurrent`.
7. Write a new subroutine called `linkedListReal_moveCurrent2Head`. This subroutine should take a `linkedListReal` as its *only* dummy argument. The subroutine should move the `current` point in the object to the head of the linked list.
8. Write a new subroutine called `linkedListReal_moveCurrent2Tail`. This subroutine should take a `linkedListReal` as its *only* dummy argument. The subroutine should move the `current` point in the object to the tail of the linked list.
9. Add a block of code to the end of the program in `linkedList04.f03` that prints the linked list

built from the input file in reverse, starting from the tail and ending at the head. The program should use the new subroutines `linkedListReal_moveCurrent2Tail`.