

## Part 1 Linked List Access (10 + 20 = 30 marks)

You are provided with the trimmed skeleton code of Assignment 2. Your job is to implement the following three features

- Access the elements in the list like an array using the operator `[]`.
- Insert an element at a certain position/index
- Remove an element at a certain position/index

### Task 1 Operator `[]` (10 marks)

Implement the operator `[]` for a List `L` such that `L[i]` will return the element of the list `L` with index `i` starting with 0 like the array. Here is an example output:

```
Operator [] Test
The linked list so far: 6 5 4 3 2 1 f e d c b a
The char with index 0 is 6
The char with index 2 is 4
The char with index 4 is 2
The char with index 6 is f
The char with index 8 is d
```

Your function should print "Index out of bound error (operator[])" if the index is out of bound.

### Task 2 Insert and Remove an item at a Certain Position (20 marks)

Implement the two functions `insertAtPos(int idx, T item)` and `removeAtPos(int idx)` that will insert or remove an item at the positive with index `idx`. Here is an example for insertion. After an insertion, the length of the list will be increased by 1.

```
Original list: f e d c b a
After
• insertAtPos(4, '#');
• insertAtPos(6, '#');
• insertAtPos(8, '+');
New list: f e d c # b # a +
```

And an example for removal. Note that the input of the function is the index, not the item. The items going to be removed are in red.

```
Original list: 6 5 4 3 2 1 f e d c b a
After
• removeAtPos(7);
• removeAtPos(3);
• removeAtPos(0);
New list: 5 4 2 1 f d c b
```

Your function should print "Index out of bound error (insertAtPos)" or "Index out of bound error (removeAtPos)" if the index is out of bound. Here is the expected input. However, if your list has `n` elements already, you can add one more item at the position with index `n`, aka, appending to the list.