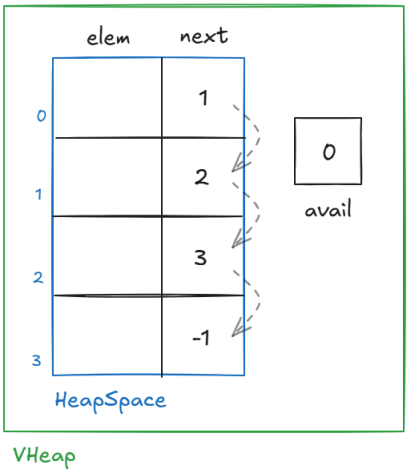
Cursor Based List

Variation 1



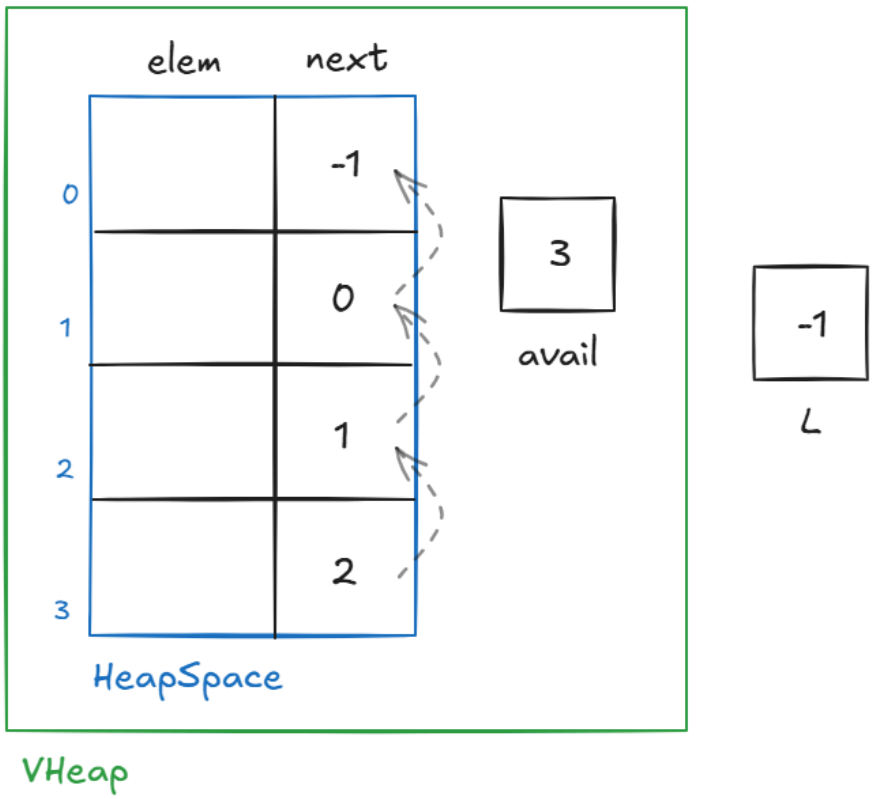

```
#define MAX 4

typedef struct {
    int elem;
    int next;
} Cell, HeapSpace[MAX];

typedef struct {
    HeapSpace H;
    int avail;
} VHeap;

typedef int List;
```

Operations	Checklist	Example
`void initialize(VHeap *V);`	☐ Set avail to the beginning of list ☐ Traverse the list and update each cell's next to create a chain ☐ Set last cell's next to -1	
`int allocSpace(VHeap* V);`	☐ Check if there is an available cell in the virtual heap ☐ Get the index of the first available cell ☐ Update avail to the next available cell ☐ Return the index of the allocated cell	
`void deallocSpace(VHeap* V, int index);`	☐ Set the next of the deallocated cell to the current avail index ☐ Update avail to point to the newly deallocated cell	

`void insertFirst(int* L, VHeap* V, int elem);`	☐ Allocate a new cell ☐ Check if allocation was successful ☐ Set the element of the new cell ☐ Set the next of the new cell to the current list head ☐ Update the list head to point to the new cell	
`void insertLast(int* L, VHeap* V, int elem);`	☐ Allocate a new cell ☐ Check if allocation was successful ☐ Set the element of the new cell ☐ Set the 'next' of the new cell to -1 ☐ Use a pointer to traverse to the last cell ☐ Update next of last cell to new cell	
`void insertPos(int* L, VHeap* V, int elem);`	☐ Allocate a new cell ☐ Set the element of the new cell ☐ Use a pointer to traverse the list ☐ Traverse to the cell before the insertion point ☐ Link the new cell into the list *Note: Cell position is NOT the same as the index of the array. Indices of the array have no relation to cell positions - they are just like addresses in linked lists. A cell's position could be 1 but its index in the array is 3.*	
`void insertSorted(int* L, VHeap* V, int elem);`	☐ Allocate a new cell ☐ Set the element of the new cell ☐ Use a pointer to traverse the list ☐ Traverse the list until you find the correct sorted position ☐ Link the new cell into the list at the correct position	
`void delete(int* L, VHeap* V, int elem);`	☐ Use a pointer to traverse to the cell with the element to delete	

	☐ Update previous cell's next to point to current cell's next ☐ Deallocate current cell	
`void deleteAllOccurrence(int* L, VHeap* V, int elem);`	☐ Traverse the list using a pointer ☐ If the current node contains the element to be deleted, unlink the node by advancing the current pointer to the next cell in the list ☐ Deallocate the removed cell ☐ Continue to traverse the list as long as there are cells	
`void display(int L, VHeap V);`	☐ Print all cell values in correct order starting from List head until next is -1	

Variation 2


elem
next
0
1
2
3
-1
0
1
2
HeapSpace
3
avail
-1
L
VHeap