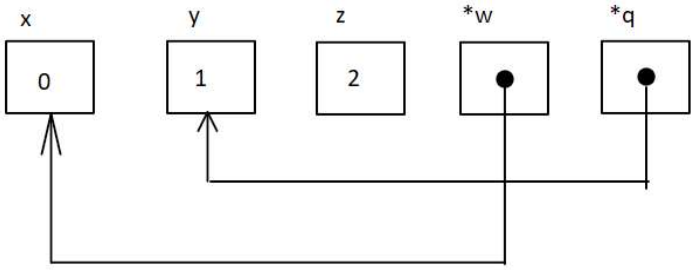
Minna og Nanna

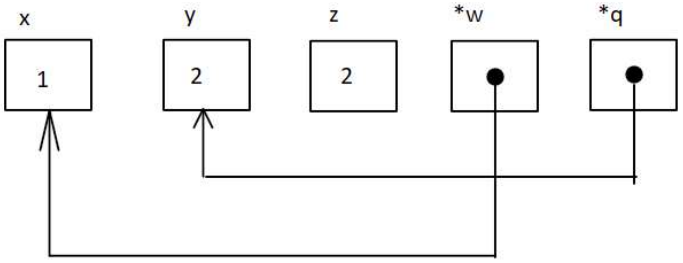
Assignment week 6

Exercise 1:
At the beginning of the program x = 0, y = 1 and z = 2, and then the pointer w is set to point to x and the pointer q to y:

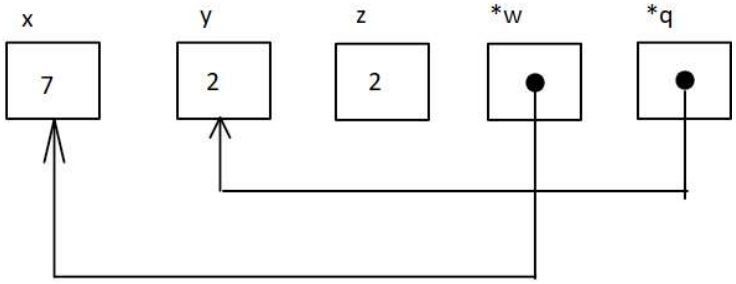
```
x=0
y=1
z=2
```



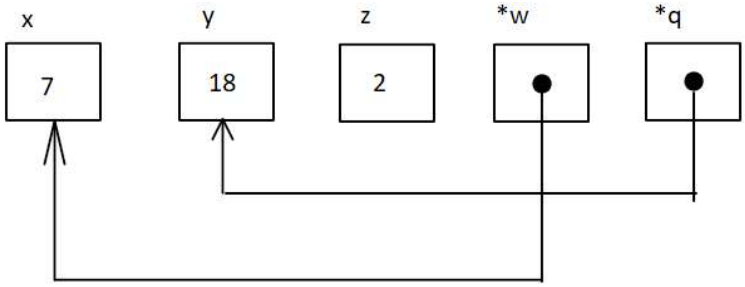
Then *w = y and *q = z, which means now the value of x = 1 and the value of y = 2:



Then the value of the pointer *w is set to be the equation *w = x + y + z + *q, so the value of *w = 7 => x = 7:



Now the value of the pointer *q is set to be the equation *w = x + y + z + *w, so the value of *q = 18 => y = 18:



The program prints “x = 7, y = 18, z = 2”.

Minna og Nanna

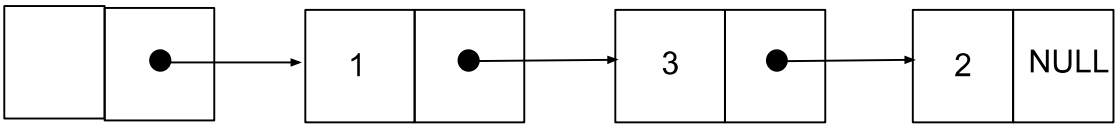
Exercise 2:

```
int max(int *numbers, int size) {
  // Exercise 2
  // Implement your code below...
  assert(size>0);
  int m = numbers[0];
  for(int i=1; i < size; ++i)
  {
    int element = numbers[i];
    if(element>m)
    {
      m=element;
    }
  }
  return m;
}
```

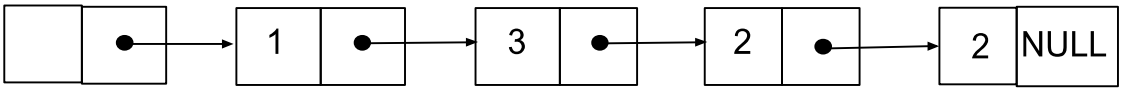
We have written a funktion, that takes an array of numbers, and returns the maximum value in the array.

Exercise 3: (Skal lige tjekkes, er lidt i tvivl om pilene)

a) The first list:



The second list:



b) We have written a function that returns the number of elements in the list.

```
// exercise 3.b
int size(node *l) {
  assert(l != NULL);
  node *p = l->next;
  int i = 0;

  while (p != NULL)
  {
    ++i;
    p = p->next;
  }
  return i;
}
```

c) If the list is not empty, p will point to the first not empty element, and run the while loop. After running the while loop, the program goes back up, and because it still is

pointing to the first not empty element, it will run the while loop with the same data again and again.
If the list is empty, so when p points to l's next, it points to null.
Because p = 0, the program will only print an empty line and will therefore not fulfill the postcondition.

d) We have corrected the function, so it fulfills the post condition.

```
// exersice 3.c and 3.d
void printout(node *l) {
  // pre:  head points to the first, empty element.
  //       The last element's next is NULL
  // post: The values of the list are printed out
  node *p = l->next;
  while (p != NULL) {
    p=p->next;
    printf("%d, ", p->data);
  }
  printf("\n");
}
```

e) We have written a function that returns the largest value of the list

```
// exercise 3.e
int largest(node *l) {
  // pre:  head points to the first, empty element.
  //       The last element's next is NULL.
  // post: Returns the largest value of the list
  node *p = l->next;
  int max = p->data;
  while(p != NULL)
  {
    if(p->data>max)
    {max=p->data;}
    p=p->next;
  }
  return max;
}
```