Programmering for computerteknologi Hand-in Assignment Exercises

Week 6: Programming with pointers

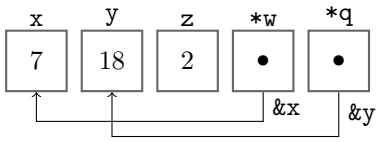
Written by: Alexander A. Christensen (202205452)

Disclaimer: Due to errors with CMake that neither me, or the TAs have solved, the test-cases have not been run. Instead, the functions have been manually tested.

The code can still be found at https://github.com/Aarhus-University-ECE/assignment-6-A-CHRI

Exercise 1

We've created 3 variables `x`, `y` and `z`, as well as two pointers `q` and `w`, which point respectively to `x` and `y`. Running the program, we can draw the following diagram.



At the end of the program, $x = 7$, $y = 18$, $z = 2$, and the pointers still point to x and y. The program then prints `x=7, y=18, z=2`.

Exercise 2

We wish to create a function that takes an array of integers, and it's size, and returns the largest integer. We do this by looping over the array, and keeping track of the largest integers index.

The code has been implemented below.

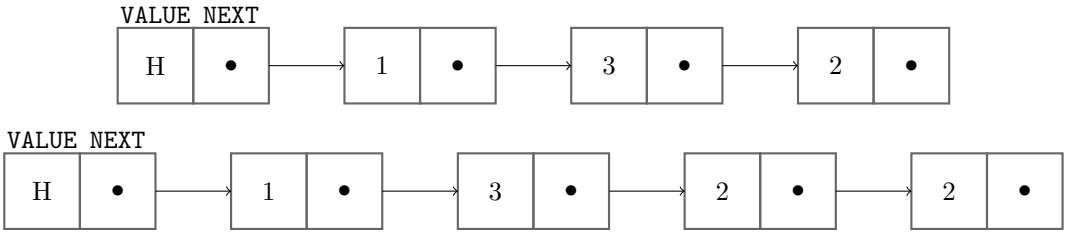
```
int max(int* numbers, int size)
{
    // Pre: size > 0
    assert(size > 0);

    int max = 0;
    for (int i = 0; i < size; i++)
    {
        if (numbers[i] > numbers[max])
            max = i;
    }

    return numbers[max];
}
```

Exercise 3

We wish to draw a diagram showing a linked list after adding various numbers. Each node in the list contains a value, as well as a pointer to the next element.



We wish to implement a function `size` that returns the size of the given list. We do this by looping over each element in the list, counting until we hit a node that point to `NULL`.

```
int size(node *l)
{
  // Excercise 3b)
  node *p = l;
  int size = 0;
  while (p->next != NULL)
  {
    size++;
    p = p->next;
  }
  return size;
}
```

We've been given the following code, and wish to describe it, and determine whether or not the code works as intended.

```
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)
  {
    printf("%d, ", p->data);
  }
  printf("\n");
}
```

The given code is supposed to take a pointer to the head of a list, and loop throughout the list, printing each element. This goes on until we hit the last node in the list.

However, the given codes postcondition does not hold, as we don't iterate throughout the list. Instead the program keeps printing the value of the 1st (non-head) node. To insure the post-condition is upheld, we need to add a line inside the while-loop to iterate to the next node.

```c
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)
  {
    printf("%d, ", p->data);
    p = p->next; /* Iterate to the next node */
  }
  printf("\n");
}
```

Lastly we wish to write a function `int largest(node *l)` which returns the largest element in a linked list. We do this by keeping track of the largest element in the list, and iteration through it.

```c
int largest(node *l)
{
    /* pre: head points to the first, empty element. The last element's next is NULL.
    size(l > 0)
    post: returns the largest value of the list */
    assert(l->next != NULL);
    node *p = l->next;
    int largest = p->data;
    while (p->next != NULL)
    {
        p = p->next;
        if (p->data > largest)
        {
            largest = p->data;
        }
    }
    return largest;
}
```