Exercise 1

Compile and run the following program:

```c
#include<stdio.h>
#include<stdlib.h>

int main() {
    int * pc;
    int c;
    c = 22;
    printf("Address of C:%d\n", &c);
    printf("Value of C:%d\n\n", c);

    pc = &c;

    printf("Value of pc:%d\n", pc);
    printf("Value stored in the memory location pointed by pc:%d\n", *pc);

    c = 11;

    printf("Value of pc:%d\n", pc);
    printf("Value stored in the memory location pointed by pc:%d\n", *pc);

    *pc = 2;

    printf("Address of c:%d\n", &c);
    printf("Value of c:%d\n", c);

    return EXIT_SUCCESS;
}
```

Exercise 2

- Write a function `bubble_sort()` which will accept an array of integers and sort it in place using **Bubble sort* algorithm**

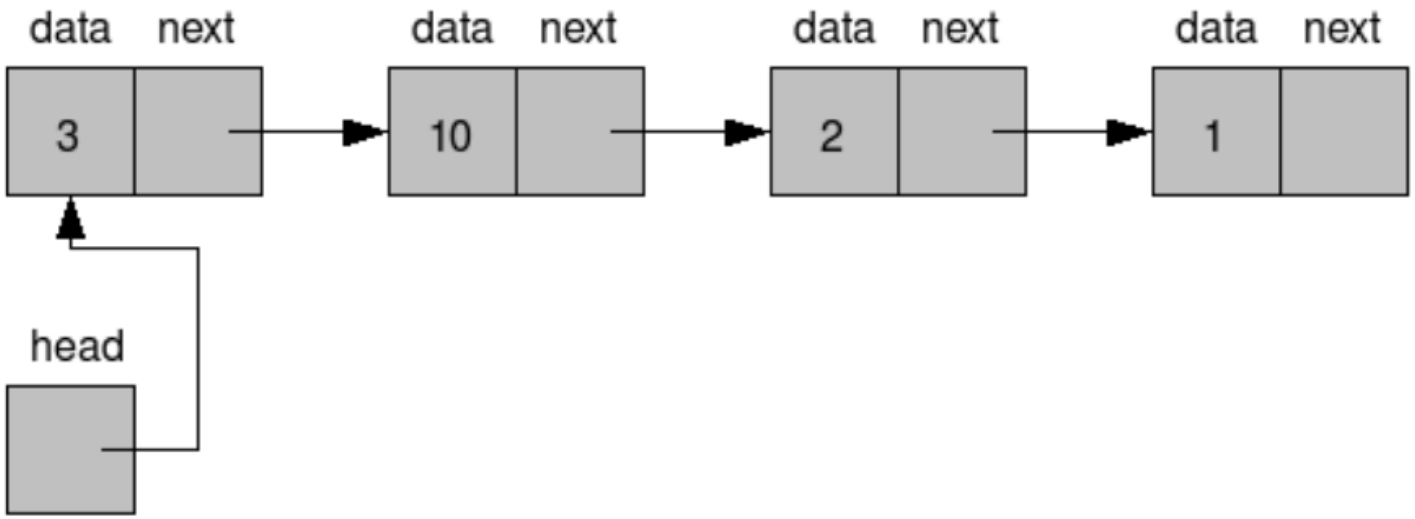
*https://en.wikipedia.org/wiki/Bubble_sort

Exercise 3 (1/2)

- Write a program that creates a `linked list*` containing integers and uses the next functions:
  - a function `print_list()` that will print out the value of each element
  - a function `insert_node()` which will insert a new element after some existing element
  - a function `delete_node()` which will delete a certain element

*https://en.wikipedia.org/wiki/Linked_list

Exercise 3 (2/2)



Linked list structure

Exercise 4 (optional)

- Implement a `Quicksort* algorithm:`
  - Pick an element, called a pivot, from the array
  - Partitioning: reorder the array so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final position. This is called the partition operation
  - Recursively apply the above steps to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values

*https://en.wikipedia.org/wiki/Quicksort

Exercise 5 (optional)

- Change your linked list implementation to a `doubly linked list*`

*https://en.wikipedia.org/wiki/Doubly_linked_list