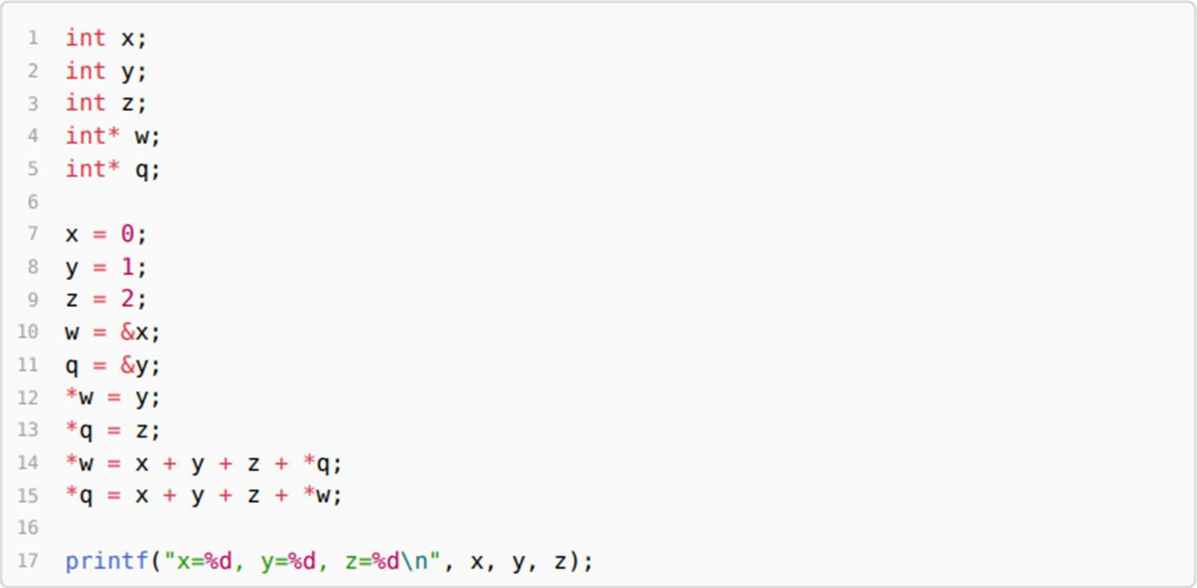
Assignment week 6

Ida Victoria Thai Thomassen & Cecilia Cvitanich Fisher CE2

Repository: https://github.com/Aarhus-University-ECE/assignment-6-idacecilia/tree/main

Exercise 1 – What does the program print when executed?

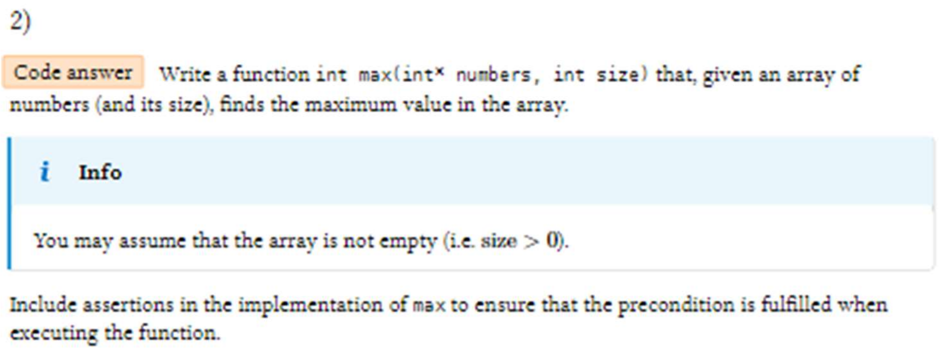


```c
int x;
int y;
int z;
int* w;
int* q;

x = 0;
y = 1;
z = 2;
w = &x;
q = &y;
*w = y;
*q = z;
*w = x + y + z + *q;
*q = x + y + z + *w;

printf("x=%d, y=%d, z=%d\n", x, y, z);
```

The program prints:

x=7, y=18, z=2

Exercise 2



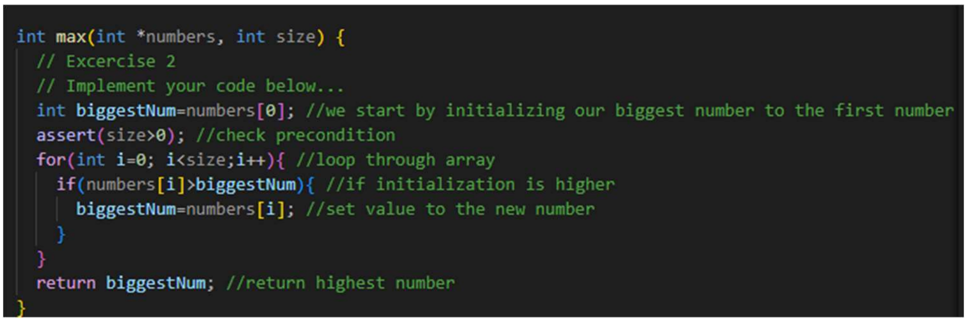
2)

Code answer Write a function `int max(int* numbers, int size)` that, given an array of numbers (and its size), finds the maximum value in the array.

i **Info**

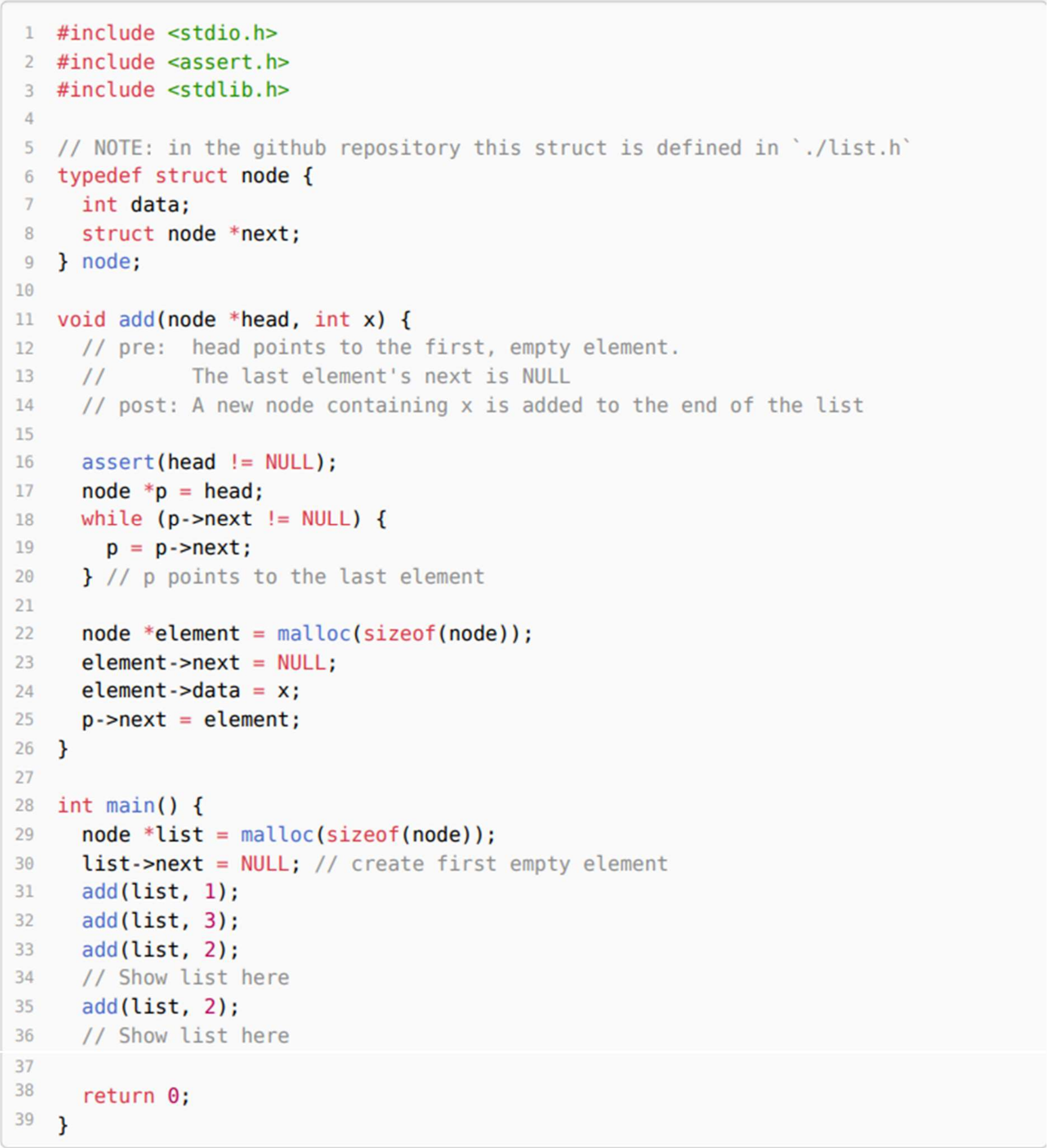
You may assume that the array is not empty (i.e. size > 0).

Include assertions in the implementation of `max` to ensure that the precondition is fulfilled when executing the function.



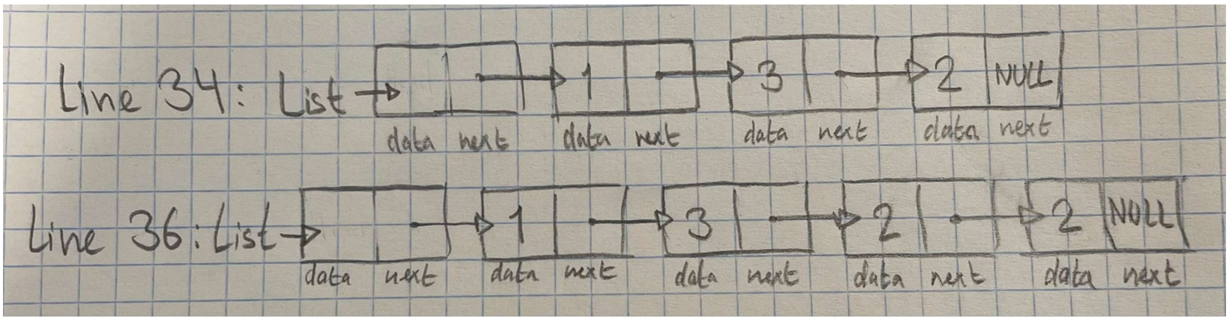
```
int max(int *numbers, int size) {
  // Excercise 2
  // Implement your code below...
  int biggestNum=numbers[0]; //we start by initializing our biggest number to the first number
  assert(size>0); //check precondition
  for(int i=0; i<size;i++){ //loop through array
    if(numbers[i]>biggestNum){ //if initialization is higher
      biggestNum=numbers[i]; //set value to the new number
    }
  }
  return biggestNum; //return highest number
}
```

Exercise 3

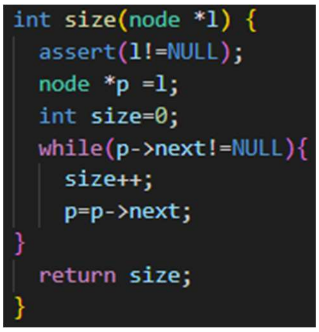


```
1   #include <stdio.h>
2   #include <assert.h>
3   #include <stdlib.h>
4
5   // NOTE: in the github repository this struct is defined in `./list.h`
6   typedef struct node {
7     int data;
8     struct node *next;
9   } node;
10
11  void add(node *head, int x) {
12    // pre:  head points to the first, empty element.
13    //       The last element's next is NULL
14    // post: A new node containing x is added to the end of the list
15
16    assert(head != NULL);
17    node *p = head;
18    while (p->next != NULL) {
19      p = p->next;
20    } // p points to the last element
21
22    node *element = malloc(sizeof(node));
23    element->next = NULL;
24    element->data = x;
25    p->next = element;
26  }
27
28  int main() {
29    node *list = malloc(sizeof(node));
30    list->next = NULL; // create first empty element
31    add(list, 1);
32    add(list, 3);
33    add(list, 2);
34    // Show list here
35    add(list, 2);
36    // Show list here
37
38    return 0;
39  }
```

a) Draw two diagrams that shows the contents of list at the two lines with the comment // show list here in main (line 34 and 36).

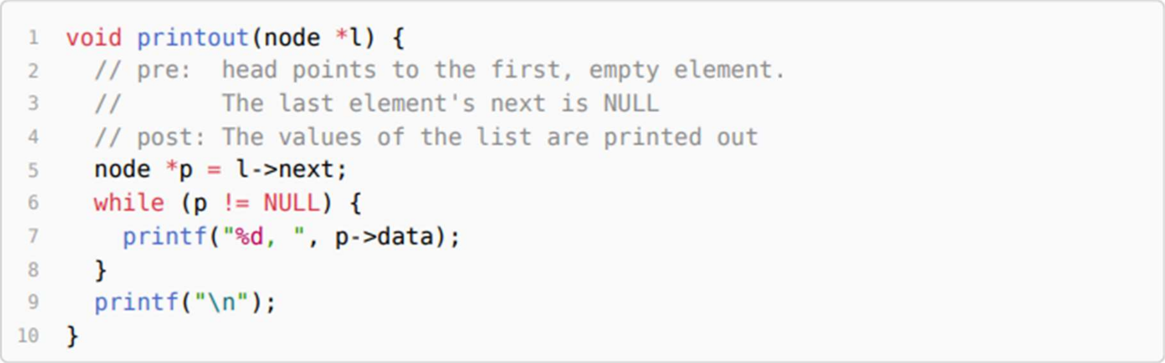


b) Implement a function with the following signature: int size(node *l). It has the same precondition as add and returns the number of elements in the list. E.g. if size(list) was printed out at the first // show list here) in main, the result would be 3.



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

c) What does the following code do when executed? (i.e. does the code fulfill the post condition? If not, what happens?)



```
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");
}
```

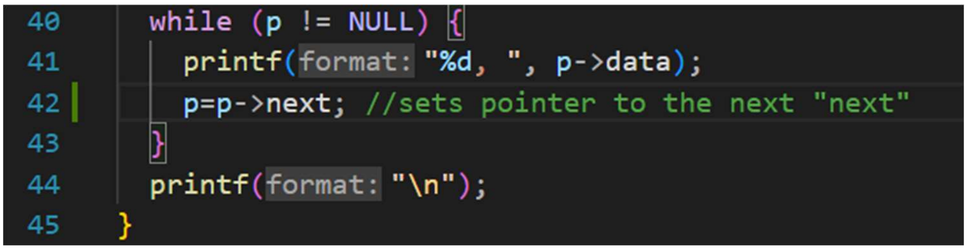
The node pointer p is set to the l->next which should point to the first non-empty element (assuming we have added elements since initializing) or the value NULL.

In case we have added elements, we enter the while loop.

The while loop will keep printing the data from the first element infinitely, since p never changes what its pointing at.

Therefore, the code does not fulfill the post condition that the values of the list are printed out.

d) Correct the function above so the post condition is fulfilled.

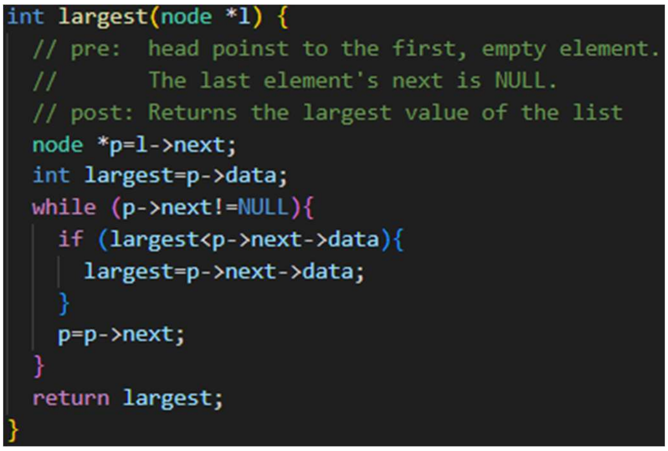


```
    while (p != NULL) {
      printf(format: "%d, ", p->data);
      p=p->next; //sets pointer to the next "next"
    }
    printf(format: "\n");
  }
```

Now we continue to the next “next” position, and it will now print every element until null occurs.

e) Write a function int largest(node *l). The pre- and post conditions are the following:

```
// pre:  head poinst to the first, empty element.
//       The last element's next is NULL.
// post: Returns the largest value of the list
```



```
int largest(node *l) {
  // pre:  head poinst 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 largest=p->data;
  while (p->next!=NULL){
    if (largest<p->next->data){
      largest=p->next->data;
    }
    p=p->next;
  }
  return largest;
}
```