Programming for computerteknologi
Hand-in Assignment Exercises

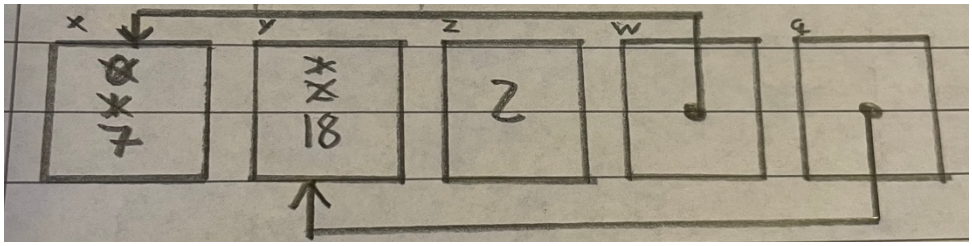
Week 6: Programming with pointers

Exercise 1)

We have been given a code to analyze. We had to calculate different equations including pointers and at the end figure out the x, y and z variable that the code prints out. The values printed out will be the following:

$x = 7,\ y = 18,\ z = 2$

I have also drawn a diagram as in the lectures to explain my answer. This diagram is included and pictured below:



Exercise 2)

We have been given a task to code a function that given an array of numbers find the greatest number in the array. The function can be seen below as well as on github.

```
5   int max(int* numbers, int size)
6   {
7       int i; /* counting variable */
8       int max = 0; /* the max value */
9
10      assert(size > 0);
11
12      for (i = 0; i < size; i++) {
13          if (numbers[i] > max) {
14              max = numbers[i];
15          }
16      }
17
18      return max;
19  }
```

When implemented in the main function it looks like this:

```c
// File for sandboxing and trying out code
int main(int argc, char **argv)
{
    int i;
    int n;

    printf("Insert length of array: \n");
    scanf("%d", &n);

    int numbers[n];

    printf("Insert numbers to go in array: \n");
    for (i = 0; i < n; i++) {
        scanf("%d", &numbers[i]);
    }

    printf("Greatest number in array: %d", max(numbers, n));

    return 0;
}
```

I couldn't get CMake to work correctly so I've been told to just post the test cases in this document. The test cases can be seen below:

```
Insert length of array:
3
Insert numbers to go in array:
76 3 2
Greatest number in array: 76
```

```c
    int numbers1[] = {5, 10, 20, 6, 10, -1, 9};
    if (max(numbers1, 7) == 20) {
        printf("5, 10, 20, 6, 10, -1, 9; Largest == 20 -> SUCCESS!\n\n");
    } else {
        printf("5, 10, 20, 6, 10, -1, 9; Largest != %d -> FAILED!\n\n", max(numbers1, 7));
    }

    int numbers2[] = {5, 1, 4};
    if (max(numbers2, 3) == 5) {
        printf("5, 1, 4; Largest == 5 -> SUCCESS!\n\n");
    } else {
        printf("5, 1, 4; Largest != %d -> FAILED!\n\n", max(numbers2, 3));
    }
```

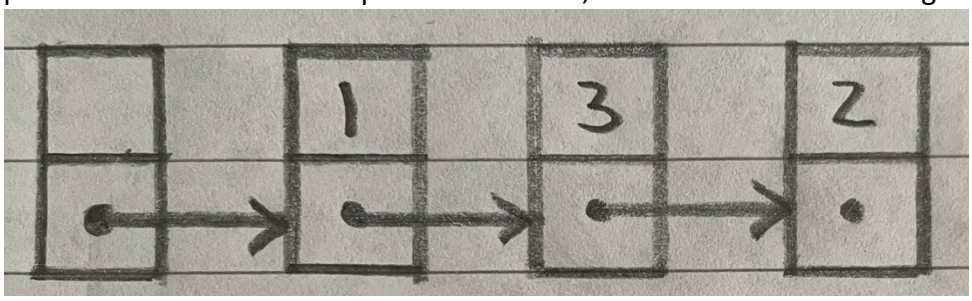
```
--------------------------
TEST CASES FROM ASSIGNMENT
--------------------------
5, 10, 20, 6, 10, -1, 9; Largest == 20 -> SUCCESS!

5, 1, 4; Largest == 5 -> SUCCESS!
```

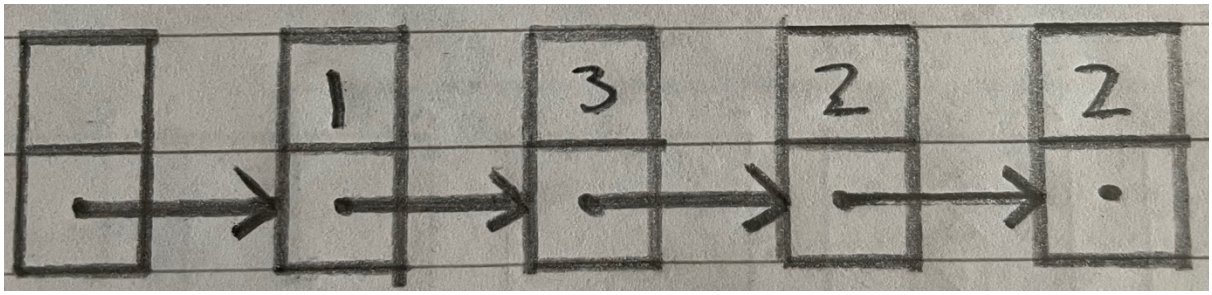
Exercise 3)

a)

We have been given a code that makes nodes with elements that can store given data. The code adds a new element to the right when running the function add(). I have made diagrams like in the lectures to show what's happening when running the code. These diagrams are included and pictured below. At the first place in the code, we will have the following diagram:



At the second place in the code, we will have this diagram:



b)
I have implemented a function which prints out the size of the list. The function can be seen below:

```
24  int size(node *l){
25    // Excercise 3b)
26    int length = 0;
27    node *p = l;
28    while (p->next!=NULL) {
29      p = p->next;
30      length++;
31    }
32
33    return length;
34  }
```

When implemented in the main function it looks like the following:

```
46      node *list = (node *) malloc(sizeof(node));
47      list->next = NULL;
48      add(list, 1);
49      add(list, 3);
50      add(list, 2);
51      add(list, 2);
52
53      printf("The length of the list is: %d", size(list));
```

When running the code the following will be printed out.

```
The length of the list is: 4
```

As I couldn't get CMake to work correctly, I've been told to just post the test cases in this document. The test cases can be seen below:

```c
    printf("\n\n-------------------------\nTEST CASES FROM ASSIGNMENT\n-------------------------\n");

    node *testl = (node *) malloc(sizeof(node));
    testl->next=NULL;
    if (size(testl) == 0) {
        printf("The length of the sequence is == 0 -> SUCCESS!\n\n");
    } else {
        printf("The length of the sequence is == %d -> FAILED!\n\n", size(testl));
    }

    testl->next = (node*) malloc(sizeof(node));
    testl->next->next=NULL;
    testl->next->data=10;
    if (size(testl) == 1) {
        printf("The length of the sequence is == 1 -> SUCCESS!\n\n");
    } else {
        printf("The length of the sequence is == %d -> FAILED!\n\n", size(testl));
    }

    testl->next->next=(node*) malloc(sizeof(node));
    testl->next->next->next = NULL;
    testl->next->next->data = 20;
    if (size(testl) == 2) {
        printf("The length of the sequence is == 2 -> SUCCESS!\n\n");
    } else {
        printf("The length of the sequence is == %d -> FAILED!\n\n", size(testl));
    }
```

```
-------------------------
TEST CASES FROM ASSIGNMENT
-------------------------
The length of the sequence is == 0 -> SUCCESS!

The length of the sequence is == 1 -> SUCCESS!

The length of the sequence is == 2 -> SUCCESS!
```

c)
We have been given an illustration of a code to analyze. The code should print out all the values of the list, but it doesn't work correctly and therefore won't fulfill the post condition. Instead, the function will print in eternity since the while loop never stops looping through. The pointer won't ever go to the next node.

d)
To correct this mistake that we discovered in the above exercise, we just must add a line that makes the pointer point to the next node, so the printout function eventually will end and therefore won't print out in eternity. The function will therefore look like the following:

```c
void printout(node *l) {
  /*Excercise 3d) Implement your changes..
    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;
  }
  printf("\n");
}
```

e)
We have been given the task to write a function, that figures out the largest value of the list. The function will look like this:

```c
int largest(node *l){
  /*Excercise 3e) Add your code below.
    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*/
  int larg = 0;
  int tempLarg = 0;
  node *p = l->next;
  while (p!=NULL){
    if (p->data > larg) {
      larg = p->data;
    }
    p = p->next;
  }
  printf("\n");
  return larg;
}
```

As I couldn't get CMake to work correctly, I've been told to just post the test cases in this document. The test cases can be seen below:

```c
    printf("\n\n-------------------------\nTEST CASES FROM ASSIGNMENT\n-------------------------\n");
    node *listTest = (node*) malloc(sizeof(node));
    node *p;
    for (int i=10; i>0; i--) {
        p = listTest;
        while (p->next!=NULL) {
            p= p->next;
        }
        p->next = (node*) malloc(sizeof(node));
        p=p->next;
        p->next=NULL;
        p->data = i;
    }
    if (largest(listTest) == 10) {
        printf("The largest value of the list is == 10 -> SUCCESS!\n\n");
    } else {
        printf("The largest value of the list is == %d -> FAILED!\n\n", largest(listTest));
    }

    listTest->next->next->next->data = 100;
    if (largest(listTest) == 100) {
        printf("The largest value of the list is == 100 -> SUCCESS!\n\n");
    } else {
        printf("The largest value of the list is == %d -> FAILED!\n\n", largest(listTest));
    }
```

```
1, 3, 2, 2,
The length of the list is: 4
The largest value of the list is: 3
```

```
-------------------------
TEST CASES FROM ASSIGNMENT
-------------------------

The largest value of the list is == 10 -> SUCCESS!


The largest value of the list is == 100 -> SUCCESS!
```