Week 6: Programming with pointers

1)

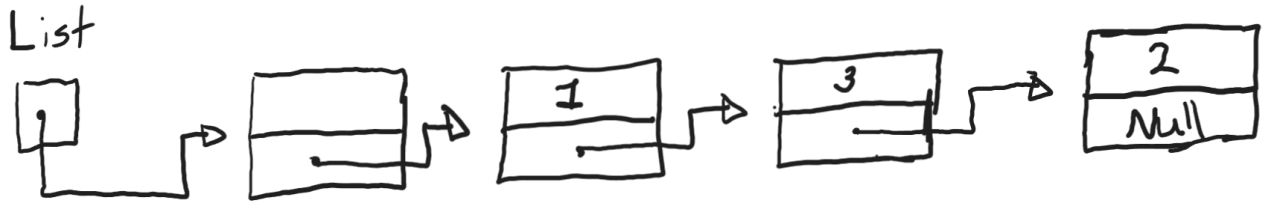
Answer: x= 7, y=18, z=2

2)

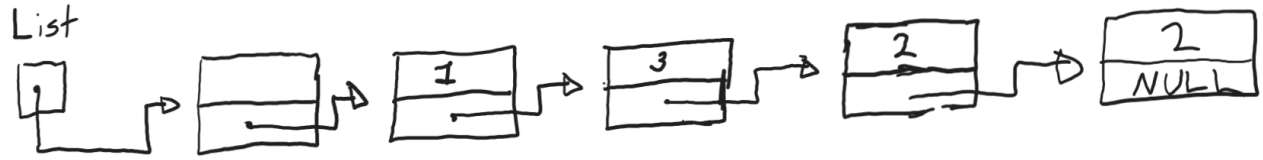
```
int max(int* numbers, int size)
{
assert(size>0);

int maxvalue = numbers[0]; /* Variable to store max value set to first place in
array */

   /* for loop to search through array to find max value */
   for (int i = 1; i < size; i++) {
        if (numbers[i] > maxvalue)
            maxvalue = numbers[i];
   }

    return maxvalue;
}
```

3)

a)



og



b)

```
int size(node *l){
    // Excercise 3b)
  node *p = l; // a pointer of type node to point at the next "next" element
  int i = 0;

  while (p->next != NULL) { //stops counting if next is NULL
    p = p->next; //updates p to be a pointer to next element
    i++; //counts size, includes first element but not last, which gives the
correct size
  }

    return i;
}
```

c)

The integer that is stored in the first not empty list will be printed forever, since pointer p is never updated and is always different from NULL, if the list is bigger than one element.

d)

```
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; // creates a pointer p that points to the first not empty
element

    while (p!=NULL){ // will print all data including the last element
      printf("%d, ",p->data); // prints data in element
      p = p->next; // updates pointer p to point to next element
    }
    printf("\n");
}
```

e)

```
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*/
    node *p = l->next; // creates a pointer p that points to the first not empty
element
    int result = p->data; //creates variable to store result and stores first
elements data there

    while(p->next!=NULL) { // runs loop until we reach last element
    if (p->data > result) {
        result = p->data; // updates result to current largest data
    }
    p = p->next; // updates p to point to next element
    }

    return result;
}
```