Question G - Daniyar Nazarbayev [H00204990].

1.

```
fun xor (x:bool) (y:bool) = (x andalso not(y)) orelse (y andalso not(x));

fun add [] [] false = false::[]

| add [] [] true = add [false] [false] true

| add (h1::t1) (h2::t2) x = (xor (xor (h1) (h2)) (x))::[]@add (t1) (t2) ((h1
andalso h2) orelse (xor (h1) (h2) andalso (x)))

| add num1 [] x = add (num1) [false] x

| add [] num2 x = add [false] num2 x;
```

I have used an extra variable for my carry. I tried to have only 2 bool lists, but it was impossible to properly carry a 1 to the next digit. The above function is basically a full adder. When calling it, use [list] [list] false as arguments.

2.

```
fun highest [] x = x

| highest (h::t) 0 = highest t h

| highest (h::t) x = if h>x then highest t h else highest t x;
```

when calling, use [list] 0 as arguments.

3.

```
fun average [] total count = total/count

| average (h::t) total count = average (t) (total+h) (count+1.0);
```

when calling, use [real list] 0.0 0.0 as arguments.

4.

```
fun incCount digit [] = (digit,1)::[]

| incCount digit ((stored_digit,number)::rest)  = if stored_digit=digit then
(stored_digit,number+1)::rest

else (stored_digit, number)::(incCount digit rest);


fun counts [] l2 = l2

| counts (hd::tl) l2 = counts(tl) (incCount (hd) (l2));


fun func [] (list,curr) = (list,curr)

| func ((number,count)::t) (list,curr) = if count=curr then func (t)
(number::list, curr) else if count>curr then func (t) (number::[],count) else
func (t) (list, curr);


fun mode list = func (counts list []) ([0],0);
```

Here are used the functions I made in exercise F (#5) – incCount and counts. They were made to take in a list, and checks how many times is something repeated in it. Then I use function named as func, that picks out the highest values, or adds them up together if they are equal. Function mode basically sticks all these functions together.