1. Buatlah fungsi gabung yang menerima dua list dan mengeluarkan sebuah list dengan tipe tuple yang merupakan gabungan dari elemen pada kedua list pada posisi yang sama. Panjang list akhir sama dengan panjang list masukan terpendek. Contoh:

```
gabung ([1;2;3;4], [1;2;3;4]) = [(1,1); (2,2); (3,3);
(4,4)]
gabung ([1;2;3;4], [1;2]) = [(1,1); (2,2)]
```

```
let rec zip = function
    |([], _) ->[]
    |(_, []) ->[]
    |(x::xs, y::ys) -> (x,y)::zip(xs, ys);;
```

2. Tentukan tipe dari ekspresi dibawah ini:

```
let rec apalah = function
  | (x::[], s) -> [s]
  | (x::xs, s) -> (s@xs)::apalah(xs, s@[x]);;
```

```
a' list * a' list -> a' list list
```

3. Diberikan fungsi sebagi berikut:

```
let rec f = function
  | ([], acc) -> acc
  | (x::xs, acc) -> f (xs, x::acc);;
```

Lengkapi evaluasi (*trace*) untuk contoh ekspresi berikut ini:

```
f ([1;2;3;4], [])
```

```
f ([1;2;3;4], [])
= f ([2;3;4], 1::[])
= f ([3;4], 2::[1])
= f ([4], 3::[2;1])
= f ([], 4::[3;2;1])
= [4;3;2;1]
```

4. Buatlah fungsi ganjilGenap yang menerima sebuah list dan mengeluarkan sebuah list dimana seluruh elemen ganjil berada di depan elemen genap. Urutan elemen pada list kembalian tidak penting. Perhatikan contoh berikut:

```
ganjilGenap [1;4;6;5;9;2] = [1;5;9;2;6;4]
ganjilGenap [1;2;7;8;4] = [1;7;8;4;2]
```

```
let rec ganjilGenap = function
     |[] -> []
     |x::xs -> if x%2=0 then (ganjilGenap xs)@[x]
                    else x::(ganjilGenap xs);;
```

5. Buatlah fungsi yang jika menerima masukan akan mengeluarkan keluaran sebagai berikut:

   entah [1;2;3] = 15

   dimana, (1*2*3) + (2*3) + 3 = 15

   entah [1;2;3;4] = 64

   dimana, (1*2*3*4) + (2*3*4) + (3*4) + 4 = 64

```
let rec entah = function
     | x::[]-> x
     | x::xs->
          let rec lah = function
               | y::[] -> y
               | y::ys -> y * lah ys
          lah (x::xs) + entah xs;;
```

6. Diberikan fungsi sebagai berikut:

```
let rec tandaTanya = function
  | ([], xacc, yacc) -> (xacc, yacc)
  | (x::xs, xacc, yacc) ->
       if (x%2=0) then tandaTanya (xs, x::xacc, yacc)
       else tandaTanya (xs, xacc, x::yacc);;
```

   Tuliskan tipe dari fungsi diatas

```
int list * int list * int list -> int list * int list
```

7. Tuliskan keluaran dari pemanggilan fungsi di bawah ini,
   `tandaTanya ([1;2;3;4;5;6;7;8], [], []);;`

```
tandaTanya ([1;2;3;4;5;6;7;8], [], []) = ([8;6;4;2], [7;5;3;1])
```

8. Buatlah sebuah fungsi yang menerima masukan sebuah list dan sebuah bilangan integer dan mengeluarkan sebuah list yang elemen-elemennya merupakan kelipatan dari n yang diambil dari list masukan. Contoh:
   `m ([2;3;6;7;1;8], 2) = [2;6;8]`

```
let rec m = function
     |([], _) -> []
     |(x::xs, t) ->
          if (x%t=0) then x::m(xs,t)
          else m(xs,t);;
```

9. Buatlah sebuah fungsi yang akan menerima masukan dan mengeluarkan keluaran sebagai berikut:
   `coba [1;2;3;4] = [1;4;2;3]`
   `coba [1;2;3;4;5;6] = [1;6;2;5;3;4]`
   `coba [4;7;2;9;1] = [4;1;7;9;2]`

```
let rec u = function
     | [] -> []
     | x::[] -> [x]
     | x::xs ->
          let rec w = function
               | (y::[], s) -> (y, s)
               | (y::ys, s) -> w (ys, s@[y])
          let (a, b) = w (xs, [])
          x::a::(u b);;
```

10. Buatlah fungsi yang akan mengeluarkan true jika jumlah elemen ganjil adalah ganjil dan jumlah elemen genap adalah genap. Jika syarat tidak terpenuhi akan mengeluarkan false. Contoh:

```
f [1;2;3;4] = false
f [1;3;8;5;2;4;6] = true
```

```
let rec c = function
     | ([], ga, ge) -> (ga%2<>0)&&(ge%2=0)
     | (x::xs, ga, ge) ->
          if x%2=0 then c (xs, ga, ge+1)
          else c (xs, ga+1, ge);;

let f xs = c (xs, 0, 0);;
```

11. Buatlah sebuah fungsi yang akan mengembalikan nilai true jika sebuah list adalah matriks.

```
isMatriks ([[1;2];[3;2]]) = true
isMatriks ([[1];[1;3]]) = false
```

```
Let rec length = function
     | ([]) -> 0
     | (x::xs) -> x+length(xs);

let rec isMatriks = function
     | ([]) -> true
     | ([x])-> true
     | (x::y::xs) -> (length(x) == length(y)) &&
isMatriks(y::xs);;
```

12. Buatlah sebuah fungsi yang dapat menghitung hasil penjumlahan dua buah matriks. Di mana ukuran list adalah dimensi dari matriks. Catatan: perlu diingat bahwa dua buah matriks tidak dapat dijumlahkan jika ukurannya tidak sama.

```
hitungMatriks ([[1;2];[3;2]], [[3;4];[6;3]]) =
[[4;6];[9;5]]
```

```
hitungMatriks ([[1];[1;3]];[[3;2];[3;3]]) = []
```

```
Let rec hitungRow = function
| ([],[]) -> []
     | ([x],[]) -> []
     | ([],[x]) -> []
     | (x::xs,y::ys) -> x+y::hitungRow(xs,ys);

let rec hitungMatriks = function
     | ([],[]) -> []
     | ([x],[]) -> []
     | ([],[x]) -> []
     | (x::xs,y::ys) -> hitungRow(x,y)::hitungMatriks(xs,ys);
```

13. Buatlah sebuah fungsi yang akan mengembalikan nilai true jika minimal ada satu pasangan angka yang memiliki hubungan panah dua arah atau simetri.

```
duaArah([(1,2);(2,4);(4,2)]) = true
duaArah([(1,2); (2,4)]) = false
```

```
Let rec cekIsi = function
     |(x, []) -> false
     |((x,y), (a,b)::As) -> if (x=b) && (y=a) then true else
cekIsi((x,y),As);

let rec duaArah = function
     | ([]) -> false
     | (x::xs) -> cekIsi(x,xs) || duaArah(xs);
```

14. Buatlah sebuah fungsi yang akan mengembalikan bentuk terbalik dari sebuah matriks.

```
reverseMatriks ([[1;2];[3;2]]) = [[2;3];[2;1]]
reverseMatriks ([[1;2;3];[3;2;4] ;[3;3;6]]) =
[[6;3;3];[4;2;3];[3;2;1]]
```

```
Let rec revList = function
     |([], rev) -> x
     |(x::xs, rev) -> revList(xs, x::rev);

Let rec reverseMatriks = function
     |([]) -> []
     |(x::xs) -> revList(x,[]):: reverseMatriks(xs);;
```

15. Buatlah sebuah fungsi yang akan mengembalikan list yang berisi tuple nilai penjumlahan dari dua buah list dan hasil pengurangannya. Hint : gunakan konsep fungsi zip yang telah diajarkan di kelas.

```
sumSubsList([1;2;6;4;3], [3;2;1;5;6]) = [(4,-2);
(4,0);(7,5);(9,-1);(9,-3)]
sumSubsList([], [2,3,9]) = [(0,-2);(0,-3);(0;-9)]
```

```
Let rec turnToMinus = function
     |([]) -> []
     |(x::xs) -> (0, 0-x)::turnToMinus(xs);

Let rec sumSubsList = function
     |([], []) -> []
     |([],x) -> sumSubsList(x)
     |(x::xs, y::ys) ->(x+y, x-y)::sumSubsList(xs,ys);;
```

16. Buatlah sebuah fungsi yang mengubah bilangan desimal ke dalam binary.

```
toBinary(10) = 1010
toBinary(20) = 10100
```

```
let rec toBinary= function
     | (0) -> "0"
     | (n) -> if n%2=0 then toBinary(n/2)+"0"
                  else toBinary(n/2)+"1";;
```

17. Buatlah sebuah fungsi yang mengembalikan sebuah list dari kelipatan k sebanyak n buah dengan nilai awal a.

```
kelipatan(k, n, a)
kelipatan(2, 5, 4) = [6;8;10;12;14]
```

```
let rec kelipatan= function
     | (k,0,a) -> []
     | (k,n,a) -> a+k::kelipatan(k, n-1, a+k);
```

18. Buatlah sebuah fungsi yang dapat mendeteksi ‘dana siluman’ yang diajukan oleh pak Camat. Output berbentuk tuple yang terdiri dari :
    a. (true, selisih harga), di mana true mengartikan bahwa barang yang diajukan termasuk ke dalam kategori ‘dana siluman’.
    b. (false, harga barang), jika barang tidak termasuk kategori ‘dana siluman’

Input dari fungsi ini adalah sebuah tuple barang yang diajukan yang terdiri dari (id_barang, harga barang) dan sebuah daftar barang yang diperbolehkan. Di dalam daftar barang yang diperbolehkan tersebut terdapat informasi id_barang dan harga sesungguhnya. Sebuah pengajuan dana dianggap sebagai 'dana siluman' jika kondisi berikut terpenuhi:

"Harga barang yang diajukan melebihi harga yang terdapat di daftar barang yang diperbolehkan."

```
cekSiluman(("a", 50000), [("a", 500);("b", 300);("c" ,
800)]) = (true, 49500)
cekSiluman(("a", 500), [("a", 500);("b", 300);("c" , 800)])
= (false, 500)
```

```
let rec cekSiluman= function
     | (x, []) -> (false, 0)
     | ((a,b), (x,y)::xs) -> if(a=x)&& (b>y) then (true, b-y)
                             elif (a=x) && ((b<y)||(b=y) then
(false, b)
                             else cekSiluman((a,b), xs);;
```

19. Buatlah sebuah fungsi yang mengembalikan semua angka di dalam list yang merupakan kelipatan dari x.

```
kelipatanX(10, [10;11;18;20;100]) = [10;20;100]
kelipatanX(3, [1;2;13;7;8;20]) = []
```

```
let rec kelipatanX= function
     | (n,[]) -> []
     | (n, x::xs) -> if x%n=0 then x::kelipatanX(xs)
                     else kelipatanX(xs);;
```

20. Lakukanlah evaluasi(trace) terhadap fungsi kelipatanX, kelipatan dan fungsi toBinary.