Task 1.

```
let rec insert (a, b) i =
if i == 0 then
   b@a
else (List.hd a)::insert (List.tl a, b) (i - 1);;
```

Task 2.

```
let rec has_element ls e =
match ls with
| [] -> false
| h::t -> if h = e then true else has_element t e;;

let rec sub_anagram fst_ls scn_ls =
match fst_ls with
| h::t -> if has_element scn_ls h then sub_anagram t scn_ls else false
| [] -> true;;

let anagram fst_ls scn_ls =
if List.length fst_ls <> List.length scn_ls then
   false
else
   sub_anagram fst_ls scn_ls;;
```

Task 3.

```
let minimize ls =
List.rev (List.tl (List.rev (List.tl ls)));;

let rec pairs ls =
if List.length ls < 2 then []
else [(List.hd ls, List.hd (List.rev ls))]@(pairs (minimize ls));;
```

Task 4.

```
let logic (a,b) c =
match c with
| 'A' -> a && b
| 'O' -> a || b
| 'X' -> a <> b
| 'I' -> not a || b
| _ -> not a;;
```