Labb kod i PDF format

UPG 1 har ej kod

UPG 2:

```
reverselist([],[]).
reverselist([H|T],Revlist):- reverselist(T,Revtail), append(Revtail,[H],Revlist).
remove_duplicates([], []).
remove_duplicates(L, E) :- reverselist(L, R), remove_duplicates2(R, R2), reverselist(R2, E).
remove_duplicates2([], []).
remove_duplicates2([H|T], NewT) :- member(H, T), remove_duplicates2(T, NewT).
remove_duplicates2([H|T], [H|NewT]) :- \+ (member(H, T)), remove_duplicates2(T, NewT).
```

UPG 3:

```
% Basfall: En tom lista har en delsträng av längd 0, vilket är den tomma listan.
delstrang(_, 0, []).
% Huvudrekursivt fall.
delstrang(Lista, L, Delstrang) :-
    % Hitta en startpunkt i listan
    append(_, Svans, Lista),
    % Ta delsträngen från startpunkten
    append(Delstrang, _, Svans),
    % Beräkna längden på delsträngen
    length(Delstrang, L),
    % Se till att L är större än 0 (eftersom vi redan har hanterat tomfall)
    L > 0.
```

UPG 4:

```
edge(a, b).
edge(a, c).
edge(b, d).
edge(c, d).
edge(d, e).
edge(e, f).
edge(c, f).
% path(Start, End, Path) - hittar en väg från Start till End och returnerar den som Path
path(Start, End, Path) :-
    traverse(Start, End, [Start], Path).
% traverse(Current, End, Visited, Path)
% - Current är den nuvarande noden
% - End är destinationen
% - Visited är en lista över redan besökta noder
% - Path är den slutliga vägen
traverse(End, End, Visited, Path) :-
    reverse(Visited, Path). % Om vi når målet, returnera den besökta vägen i rätt ordning
traverse(Current, End, Visited, Path) :-
    edge(Current, Next),     % Hitta en kant från Current till en ny nod Next
    \+ member(Next, Visited), % Se till att Next inte redan är besökt
    traverse(Next, End, [Next | Visited], Path). % Fortsätt traverseringen
```