Erlang-mitirki

Az alábbi "mitirki"-feladatokban az Erlang BIF-ben és STDLIB-ben definiált függvények közül az itt felsoroltakat használjuk. Definíciójukat lásd itt:

```
BIF/ERTS -- <http://erlang.org/doc/apps/erts/index.html>,
STDLIB -- <http://erlang.org/doc/apps/stdlib/index.html>

element/2
hd/1
length/1
not/1
round/1
size/1
tl/1

erlang:'*'/2
erlang:'+'/2
erlang:'++'/2
erlang:'-'/2
erlang:'--'/2
erlang:'hd'/1
erlang:'tl'/1
erlang:element/2
erlang:not/1
erlang:round/1
erlang:size/1

lists:concat/1
lists:filter/2
lists:foldl/3
lists:foldr/3
lists:last/1
lists:map/2
lists:partition/2
lists:reverse/1
lists:seq/2
lists:split/1
lists:sublist/2

string:substr/3
```

Az alábbi "mitirki"-feladatokban a következő karaktervizsgáló függvényeket használjuk. A karaktervizsgáló függvények modulnév megadása nélkül hívhatók.

```
is_space(X) ->
    X =:= 32.

is_not_space(X) ->
    X =/= 32.

is_num(X) ->
    X >= $0 andalso X =< $9.

is_alpha(X) ->
    X >= $A andalso X =< $Z orelse X >= $a andalso X =< $z.

is_alphanum(X) ->
    is_alpha(X) orelse is_num(X).

is_not_alpha(X) ->
    X < $A orelse X > $Z andalso X < $a orelse X > $z.
```

Mi lesz az X változó értéke az alábbi kifejezések kiértékelésekor?
Az egyes deklarációk függetlenek egymástól.
A karaktervizsgáló függvények modulnév megadása nélkül hívhatók.

Mivel az Erlang (az 'andalso' és 'orelse' operátorok kivételével) mohó kiértékelésű, a kifejezéseket addig egyszerűsíti, ameddig csak lehet. X értékeként ezt a lehető legegyszerűbb, ún. kanonius kifejezést várjuk.

Gondoljon arra is, hogy ha egy egészlista csak nyomtatható karakterek ASCII-kódját tartalmazza, ide értve a szóközt, az új sort, a tabulátort és más vezérlő karaktereket, akkor ezt a listát az Erlang értelmező sztringként -- másnéven füzérként, karakterláncként -- írja ki.

```
 1. X = {(fun erlang:'--'/2)("mama","alma"), 4-3, [$R,$S,$T]}.
 2. F = fun(L) -> lists:filter(fun is_not_space/1, L) end, X = F([$a,$ ,$ ,$c]).
 3. F = fun lists:seq/2, X = [{X,Y} || X <- F(1, 3), Y <- F(4, 5), X*Y > 6].
 4. X = {("kor"++"let")--"teke", (fun erlang:'*'/2)(3, 4), true =:= false}.
 5. F = fun(L) -> lists:filter(fun is_alphanum/1, L) end,
    X = F([0,$m,2,$a,$0,c,1000]).
 6. L = lists:seq(2, 4),
    X = [{X,Y,Z} || X <- L, Y <- L, Z <- L, X*Y-Y*Z =:= X*Z].
 7. X = {round(3.4-4.3), (fun erlang:'++'/2)([1], [2]), [""]}.
 8. F = fun(L) -> lists:filter(fun is_alpha/1, L) end, X = F([$a,$9]++"dp2018").
 9. F = fun lists:seq/2,
    X = [F(X, Y) || X <- lists:reverse(F(1, 3)), Y <- F(2, 3), X =< Y].
10. X = {[lists:reverse("al")]++["ma"], round(fun erlang:'-'/2(3.4, 4.3)),
         [[3 =:= 2*2]]}.
11. F = fun(L) -> lists:map(fun is_num/1, L) end, X = F([1,$1,x,$x]).
12. F = fun lists:seq/2,  X = [X+Y || X <- F(3, 4), Y <- F(3, 5), X*Y < 17].
13. X = {$9-$7, "4*4", string:substr("4*4", 1, 2), [4 < 4.2]}.
14. F = fun(L) -> lists:map(fun erlang:'not'/1, L) end,
    X = F([a=:=a,a=:=b,true,false]).
15. L = lists:seq(2, 4), X = [X+Y-Z || X <- L, Y <- L, Z <- L, 2*X+3*Y =:= Z*Z].
16. X = {[$3,$4]++"2",'.2', string:substr("2.3", 1, 2), (2 > 2)}.
17. X = lists:filter(fun(X) -> element(2, X) end,
                     [{true,true},{true,false},{false,true},{1 > 2, 2 > 1}]).
18. F = fun lists:seq/2,
    X = [(X+Y)-(X-Y) || X <- F(1, 3), Y <- F(2, 5), X*Y < 8].
19. X = {[$a+3], size({"2",12,a}), round(2.22*3)}.
20. X = lists:partition(fun(L) -> length(L) =< 3 end,
                        [[],[4,6,7,3],[3,3,3],[a,a],"kukacos"]).
21. F = fun lists:seq/2, X = [$a+X || X <- F(0, 2), Y <- F(1, 3), X+Y > 2].
22. X = {round(2.3-5.1), length("2.3-5.4"), {"2.3-5.4", -56}}.
23. X = lists:map(fun({F,X}) -> F(X) end,
                  [{fun erlang:'size'/1, {3,1}},
                   {fun lists:last/1, [1,2,5]}, {fun erlang:'not'/1, 1<1}]).
24. F = fun lists:seq/2, X = [$a+Y || X <- F(0, 2), Y <- F(0, 3), X+Y < 3].
25. X = {hd(tl([3,2,4])), element(3, {1,"1.0",1.0}), is_space($3)}.
26. X = lists:map(fun({F,X}) -> {X,F(X)} end,
                  [{fun erlang:'tl'/1,[3,1,2]},{fun 'length'/1,[]}]).
```

```
27. L = lists:seq(1, 7), X = lists:split(4, [X+Y || X <- L, Y <- L, X*Y < 5]).

28. X = {"12*3", round(12.0*3.0), (12*3.0)}.

29. X = lists:concat(lists:map(fun(X) -> element(2, X) end,
                               [{a,b},{d,e},{f,g}])).

30. F = fun lists:seq/2, X = [X+Y || X <- F(2, 4), Y <- F(4, 6), X*Y < 12].

31. X = {"2", $3-$0, {"4X"++[$U], tl([5,3,0])}}.

32. X = lists:map(fun(X) -> X+2 end, [$b,$2]++"c3").

33. L = lists:seq(3, 5), X = [X+Y-Z || X <- L, Y <- L, Z <- L, X*Y+Z =:= X*Z].

34. X = {"Mi"++"lord", [$C+3], 5 =/= 2, trunc(21.3)}.

35. X = lists:foldl(fun erlang:'=:='/2, 2 =:= 4,
                    [1.9 < 1.99, true, "A" < "ABA"]).

36. F = fun lists:seq/2, X = [$A+X || X <- F(2, 3), Y <- F(0, 3), X+Y < 4].

37. X = {{2*3.0,3*2}, element(2,{2,3.0,e}), [$a+2]}.

38. X = lists:filter(fun is_alpha/1, lists:concat(["Bp.","21. ker.",""])).

39. F = fun lists:seq/2, X = [$a+Y || X <- F(2, 3), Y <- F(0, 3), X+Y < 4].

30. X = {length("length"), size({"Size"}), (fun(X) -> X < 4 end)(4)}.

41. X = lists:filter(fun is_num/1, lists:concat(["41a5","x","8z"])).

42. L = lists:seq(2, 10),
    X = lists:sublist([X+Y || X <- L, Y <- L, X*Y < 12], 3).

43. X = {{some,6}, [$A], length([fun erlang:'element'/2]),
         (fun erlang:'>'/2)(2,3)}.

44. X = lists:map(fun is_not_space/1, lists:concat(["x 7","a5 ","8 z"])).

45. F = fun lists:seq/2,
        X = [F(X, Y) || X <- F(1, 2), Y <- lists:reverse(F(1, 3)), X =< Y].

46. X = {"321.0", (321.0/321.0), lists:reverse("1"++"2")++"3"}.

47. X = lists:map(fun(S) -> tl(S) end, ["4a5b","like","675"]).

48. F = fun lists:seq/2, X = [$h+X || X <- F(0, 2), Y <- F(0, 2), X+Y < 3].

49. X = {[$a]++[$a+1]++[$a+2], [1,2|[3]], {5/1,4}}.

50. X = lists:foldr(fun(X,Xs) -> [X|Xs] end, [round(3.2)], [3,4-2]).

51. F = fun lists:seq/2,
    X = [{X,Y*Y} || X <- F(1, 3), Y <- F(3, 4), X*X*Y*Y < 40].

52. X = {[tl("ex")], "jo"++[1+$a,$b], {7/2}}.

53. X = lists:filter(fun(X) -> fun is_not_alpha/1(X) end, "a23").

54. L = lists:seq(3, 7), X = lists:split(1, [X+Y || X <- L, Y <- L, X*Y < 10]).

55. X = {["bak"++tl(tl("agancs"))], 5*2-3, {0,[$a]++"bc"}}.

56. X = lists:filter(fun(L) -> length(L) =< 3 end,
                     [[],[4,6,7,3],[3,3],[3,3,3]]).

57. F = fun lists:seq/2,
    X =[F(X, Y) || X <- F(1, 4), Y <- F(2, 3), X =< Y, Y-X < 2].

58. X = {tl("hex")++"em", [5.0-2, 5-2], {y,"y",'Y'}}.

59. X = lists:map(fun(X) -> $m+X end, [-2,-4,5])++"ály!".

60. F = fun lists:seq/2,
    X = [F(Y, X) || X <- F(2, 3), Y <- F(2, 4), X >= Y, X-Y < 4].
```