```
> #Lab 9 (Fri) solutions
>
> #Q1.
> power2 :=proc(n)
   if n < 1 then return false;
   elif  n = 1 then return true;
   elif type(n, integer) = false then return false;
   else return power2(n/2);
   end if;
   end proc;
power2 := proc(n)                                                          (1)
    if n < 1 then
        return false
    elif n = 1 then
        return true
    elif type(n, integer) = false then
        return false
    else
        return power2(1/2*n)
    end if
end proc
>
> power2(6);
                              false                                        (2)
> power2(32);
                              true                                         (3)
> power2(16);
                              true                                         (4)
> power2(-2);
                              false                                        (5)
>
> #Q2
> rechf :=proc(n :: posint)
   if  n = 1 then return 1;
   else return n^n·rechf(n-1);
   end if;
   end proc;
rechf := proc(n::posint)                                                   (6)
    if n = 1 then return 1 else return n^n * rechf(n − 1) end if
end proc
>
> rechf(2);
                                4                                          (7)
```

```
> rechf(3);
                                  108                                      (8)
> rechf(16);
14922760980383708438153873295130345243552690206970449281811392985992081714464\   (9)
    816585406145840471905409171456000000000000000000000000000000
>
>
>
> #Q3
> q3proc :=proc(L :: list)
   local LL;
  if nops(L) ≤ 2 then return L[1];
   else
   LL := L[3 ..nops(L)];
   return L[1] + q3proc(LL);
   end if;
   end proc;
q3proc := proc(L::list)                                                    (10)
    local LL;
    if nops(L) <= 2 then
        return L[1]
    else
        LL := L[3 ..nops(L)]; return L[1] + q3proc(LL)
    end if
end proc
>
> q3proc([1, 2, 3, 4, 5, 6, 7]);
                                  16                                       (11)
> q3proc([1.1, 2, 3, 4, 5, 6, 7]);
                                 16.1                                      (12)
>
>
> #Q4
> q4proc :=proc(L :: list)
   local LL :
   if frac(nops(L)/2) ≠ 0 then print("Error: odd length list"); return 0;
   elif nops(L) = 2 then return L[1]^L[2] ;
   else
   LL := L[3 ..nops(L)];
   return L[1]^L[2]·q4proc(LL);
   end if;
   end proc;
q4proc := proc(L::list)                                                    (13)
    local LL;
    if frac(1/2*nops(L)) <> 0 then
        print("Error: odd length list"); return 0
    elif nops(L) = 2 then
        return L[1]^L[2]
    else
        LL := L[3..nops(L)]; return L[1]^L[2]*q4proc(LL)
    end if
end proc
>
> q4proc([2, 2, 1, 2, 3, 2]);
                                   36                                   (14)
>
```