```
> #Lab 6 Thu solutions
>
> #Q1.
> oddList :=proc(L :: list)
  local num, L2;
  L2 := [ ];
  for num from 1 to nops(L) by 2 do
  L2 := [op(L2), L[num]];
  end do;
  return L2;
  end proc;
```

```
oddList := proc(L::list)                                                    (1)
    local num, L2;
    L2 := [ ]; for num by 2 to nops(L) do L2 := [op(L2), L[num]] end do; return L2
end proc
```

```
> oddList([1, 2, 3, 4, 5, 6, 7]);
                                  [1, 3, 5, 7]                              (2)
>
> #Q2
> q2 :=proc(L :: list)
  local num, L2;
  L2 := [ ];
  for num from 2 to nops(L) do
  if L[num] -L[num − 1] > 1 then L2 := [op(L2), num];
  end if;
  end do;
  return L2;
  end proc;
```

```
q2 := proc(L::list)                                                         (3)
    local num, L2;
    L2 := [ ];
    for num from 2 to nops(L) do
        if 1 < L[num] − L[num − 1] then L2 := [op(L2), num] end if
    end do;
    return L2
end proc
```

```
> q2([1, 2, 3, 5, 7, 8]);
                                  [4, 5]                                    (4)
>
> #Q3
> outlier :=proc(L :: list)
  local num, L2, sum, avge;
   sum := 0;
  for num from 1 to nops(L) do
  sum := sum + L[num];
  end do;
```

```
    avge := sum/nops(L);
    L2 := [ ];
    for num from 1 to nops(L) do
    if L[num] < 0.5·avge or L[num] > 1.5·avge then L2 := [op(L2), L[num]];
    end if;
    end do;
    return L2;
    end proc;
outlier := proc(L::list)                                                    (5)
    local num, L2, sum, avge;
    sum := 0;
    for num to nops(L) do sum := sum + L[num] end do;
    avge := sum / nops(L);
    L2 := [ ];
    for num to nops(L) do
        if L[num] < 0.5 * avge or 1.5 * avge < L[num] then L2 := [op(L2), L[num]] end if
    end do;
    return L2
end proc
> outlier([1, 2, 3, 6]);
                                    [1, 6]                                  (6)
> #Q4
> mysqlist :=proc(L :: list)
    local num, result, halflen;
     result := true;
    if frac(nops(L)/2) ≠ 0 or L = [ ] then return false;
     else
    halflen := nops(L)/2;
    for num from 1 to halflen do
    if L[num] ≠ L[num + halflen] then result := false;
    end if;
    end do;
    end if;

    return result;
    end proc;
mysqlist := proc(L::list)                                                   (7)
    local num, result, halflen;
    result := true;
    if frac(1/2 * nops(L)) <> 0 or L = [ ] then
        return false
    else
        halflen := 1/2 * nops(L);
```

```
        for num to halflen do
            if L[num] <> L[num + halflen] then result := false end if
        end do
    end if;
    return result
end proc
```

```
> mysqlist([1, 2, 3, 1, 2, 3]);
```

$$true \quad (8)$$

```
> mysqlist([1, 2, 3, 2, 3, 3]);
```

$$false \quad (9)$$

```
> mysqlist([ ]);
```

$$false \quad (10)$$

```
>
> #Q5
> listcopy :=proc(L :: list, n :: posint)
    local num, L2;
    L2 := [ ];
    for num from 1 to n do
    L2 := [op(L2), op(L)];
    end do;
    return L2
    end proc;
```

$$listcopy := \mathbf{proc}(L::list, n::posint) \quad (11)$$

```
    local num, L2;
    L2 := [ ]; for num to n do L2 := [op(L2), op(L)] end do; return L2
end proc
```

```
>
> listcopy([1, 2, 4], 3);
```

$$[1, 2, 4, 1, 2, 4, 1, 2, 4] \quad (12)$$

```
>
```

(13)