Exercise 1.

For I = 5, sum all array with scan over the array from first position until last position (step 1) faster than use step 2. Because when system uses method as step 1, system only sums the entire array with doing increment number of array, whereas method on step 2, system should checking value for each array before doing sum its make we got waiting time.

This is source code for exercise 1:

```
procedure TForm1.RandomsClick(Sender: TObject);
var
 x,y, sum1, sum2, h, j, z : int64;
 i:integer;
 str, Timestart1, Timestart2, TimeEnd1, TimeEnd2 :string;
 A : array of integer;
 DateTime : TDateTime;
begin
  DateTime := Time;  // store the current date and time
   // convert the time into a string
   randomize;
   x:=10000;
   y:=random(x);
   sum1:=0;
   SetLength(A,y);
   Timestart1 := TimeToStr(DateTime);
for i:=1 to y do
  begin
    str:=''+inttostr(i) +str;
    A[i]:=i;
    sum1:=A[i]+sum1;
  end;
  Result.Text:=inttostr(sum1);
  TimeEnd2 := TimeToStr(DateTime);
```

```
begin
Timestart2 := TimeToStr(DateTime);
sum2:=0;
for i:=1 to y do
  begin
    repeat
          j:=random(y);
             if A[j]=0 then
                begin
                  A[j]:=i;
                  z:=1;
                end
             else
                  z:=0;
     until z>0;
   end;
  h:=1;
  for i:=1 to length(A) do
      begin
        sum2:=A[h]+sum2;
        h:=A[h];
      end;
   Result1.Text:=inttostr(sum2);
   TimeEnd2 := TimeToStr(DateTime);
end;
end;
end.
```

Exercise 2

A code with code words {1, 11, 12} has the prefix property; a code consisting of {1, 11, 12} does not, because "1" is a prefix of both "11" and "12".

In Elias Code, all code will be write in binary and give value in front of the binary with zero times.

Code {1,11,12} will be like {1,0001011,001100}, and to read this code, we use method : " read bit after 1 with n times zero before bit 1.

Ex, for 0001011 : we found 3 times zero, so we can get code 1011 (read 3 bit after 1), and 1011 its means 11.

That method makes Elias code became Prefix-free, because in elias code we can read all code without ambiguous value and prefix number.

For Elias Delta code is also Prefix-free because we never got prefix number in the code, to write down the code, we use this method:

- Write it in binary.
- Count the bits and write down that number of bits in binary (X).
- Use the binary digit written in step 1 again, remove the leading bit and write down the remaining bits (Y).
- Append the second binary digit (Y) to the first binary digit (X).
- Count the bits written in step 2 (X), subtract 1 from that number and prepend that many zeros.

Example for integer 17, we can write with Elias delta code like this:

$17=2^4+1$, For N' = 4 and for N = 5, it will be 001010001.

Exercise3

Exercise 4.

Its more faster to decompress list than read list from uncompressed.

This source code for exercise 4.

```
unit exercise4_4;


interface


uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;


type
  TForm1 = class(TForm)
    Exit: TButton;
    Randoms: TButton;
    Decode: TButton;
    RandomInteger: TEdit;
    LongElias: TEdit;
    ListOfDoc: TMemo;
    EliasPrefix: TMemo;
    DecodeList: TMemo;
    procedure ExitClick(Sender: TObject);
    procedure RandomsClick(Sender: TObject);
    procedure DecodeClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
```

```
  end;

var
  Form1: TForm1;
   Adata: array of integer;
   resulty,str:string;
   binary:integer;
implementation
{$R *.dfm}

function BinToInt(str1: string): integer;
var
  c: integer;
  len: integer;
  ch: char;  // one character of the string
begin
     binary:=0;
     len:=length(str1);
     for c:=len downto 1 do
     begin
          ch:=str1[c];
          if ch='1' then
             binary:=binary+1 shl (len-c);
             // basically result:=result+(2 to the power of (len-c));
             // len-c is the digit of the number
     end;
end;

function IntToBin(i: integer): string;
var
```

```
  d: integer; // digit
  v: byte; // digit value
  n: boolean;
begin
     n:=(i<0);
     i:=abs(i);// make it positive
     d:=0;
     resulty:='';
     repeat
     begin
          v:=(i shr d) and 1; // v = 0..1
          resulty:=chr(v+48)+resulty;
          inc(d);
     end;
     until (i<(1 shl d));
     if n then
        resulty:='-'+resulty;
end;

procedure QuickSort(var A: array of Integer; iLo, iHi: Integer) ;
var
Lo, Hi, Pivot, T: Integer;
begin
  Lo := iLo;
  Hi := iHi;
  Pivot := A[(Lo + Hi) div 2];
 repeat
    while A[Lo] < Pivot do Inc(Lo) ;
      while A[Hi] > Pivot do Dec(Hi) ;
       if Lo <= Hi then
```

```
        begin
          T := A[Lo];
          A[Lo] := A[Hi];
          A[Hi] := T;
          Inc(Lo) ;
          Dec(Hi) ;

        end;
 until Lo > Hi;
    if Hi > iLo then QuickSort(A, iLo, Hi) ;
    if Lo < iHi then QuickSort(A, Lo, iHi) ;
end;

procedure TForm1.ExitClick(Sender: TObject);
begin
application.Terminate;
end;

procedure TForm1.RandomsClick(Sender: TObject);

var
x, y, integers, max, min, long :int64;
str, listDoc, insertzero, allresult, all :string;
i,j:integer;


begin
Randomize;

i:=0;
```

```
 {this is will give us always postive random list from integer (never minus)}
repeat
  integers:=9223372036854775807;
  x:=abs(random(integers)); {randomize integer}
  y:=x+x;
 if (y=0) then
   begin
     x:=random(integers);
     end
 else
   begin
    i:=1;
    end;
until i>0;
  Longelias.Text:='';
  str:=trim(inttostr(x));
  max:=strtoint(str[1]);
  min:=strtoint(str[2]);

  SetLength(AData,length(str));

for i:= 1 to length(str) do
   begin
      Adata[i]:=strtoint(str[i]);
      // find minimum value and maximum value
      if (strtoint(str[i])>= min)then
         min:=min
      else
         min:=strtoint((str[i]));
      if strtoint(str[i])<= max then
```

```
         max:=max
      else
         max:=strtoint(str[i]);
   end;

 QuickSort(Adata,min,max); //called function quicksort to sorting the array

      for i := 1 to (length(str)) do
        begin
          inttobin(Adata[i]);    {call  fungtion  inttobin,  to  convert  from
integer to binary}
            long:=length(resulty)-1; {count how many zero we need}
            for j:=1 to long do      {insert zero n}
                begin
                  insertzero:='0'+insertzero;
                end;
            allresult:=insertzero+resulty;
            resulty:=inttostr(0);
            insertzero:=''; // normalize value
            all:=all+allresult;
          Longelias.Text:=all;
          EliasPrefix.Lines.Insert(i,allresult); {write down result for Elias
prefix free}
          listDoc:='doc'+inttostr(Adata[i]);
          ListOfDoc.Lines.Insert(i,listdoc);  {write  down  the  result  from
uncompressed}
          RandomInteger.Text:=inttostr(x);
        End;
end;
procedure TForm1.DecodeClick(Sender: TObject);
var
str,str1: string;
```

```
i,j,zero: integer;
stop:boolean;
begin
  {decode to be an integer}
    zero:=0;
    stop:=false;
    for i:= 1 to EliasPrefix.Lines.Count-1 do

       begin
         str:=trim(EliasPrefix.lines.strings[i]);
          for j:=0 to length(str) do
              begin
                if (str[j]='0') and (stop = false)then
                   begin
                     inc(zero);
                     stop:=false;
                   end
                else
                 if (str[j]='1') and (stop=false)then
                   begin
                    str1:=copy(str, j, zero);
                    stop:=true;
                   end;
              end;
             bintoint(str1);   {call function coverter}
             DecodeList.Lines.Insert(i,inttostr(binary));
            stop:=false;
       end;
end;
end.
```