LAMPIRAN A

LISTING PROGRAM

Arithmetic Dynamic Bitwise

```
Private OutStream() As Byte
Private OutPos As Long
Private OutBitCount As Integer
Private OutByteBuf As Byte
Private Const MaxBits = 24  'at 16 bits precision is not high enough


Public Sub Compress_ArithMetic_DMC(ByteArray() As Byte)
    Dim InpPos As Long
    Dim LowValue As Long
    Dim HighValue As Long
    Dim RangValue As Long
    Dim MidValue As Long
    Dim Char As Byte
    Dim X As Integer
    Dim Bitset As Integer
    Dim Index As Integer
    Dim TopBit As Long
    Dim One(256) As Long
    Dim Zero(256) As Long
    Call Init_Ari_Bit2
    LowValue = 0
    HighValue = (2 ^ MaxBits) - 1
    TopBit = 2 ^ (MaxBits - 1)
    InpPos = 0
    Index = -1
    Debug.Print HighValue, TopBit


    For X = 0 To 256
       One(X) = 1
       Zero(X) = 1
    Next
    Do
       If InpPos > UBound(ByteArray) Then
          Exit Do
       Else
          Char = ByteArray(InpPos)
          InpPos = InpPos + 1
       End If
       For X = 0 To 7
          Bitset = (Char And (2 ^ (7 - X))) And &HFF
          Index = (1 * (2 ^ X)) - 1 + Int(Char / (2 ^ (8 - X)))
          RangValue = HighValue - LowValue
          MidValue = LowValue + (RangValue * (Zero(Index) / (One(Index) + Zero(Index))))
          If MidValue = LowValue Then MidValue = MidValue + 1
          If MidValue = HighValue - 1 Then MidValue = MidValue - 1
          If Bitset > 0 Then
             LowValue = MidValue
             One(Index) = One(Index) + 1
          Else
```

```
            HighValue = MidValue

            Zero(Index) = Zero(Index) + 1

        End If

        If AritmaticRescale = True Then

            If One(Index) > 127 Or Zero(Index)
> 127 Then

                One(Index) = Int(One(Index) / 2)
+ 1

                Zero(Index) = Int(Zero(Index) / 2)
+ 1

            End If

        End If

        Do While (HighValue And TopBit) =
(LowValue And TopBit) Or LowValue >
HighValue - 255

            If (LowValue And TopBit) = 0 Then

                Call AddBitsToOutStream(0, 1)

            Else

                Call AddBitsToOutStream(1, 1)

            End If

            HighValue = (HighValue And
(TopBit - 1)) * 2 + 1

            LowValue = (LowValue And
(TopBit - 1)) * 2

            If LowValue >= HighValue Then
HighValue = (2 ^ MaxBits) - 1

        Loop


    Next

Loop

For X = MaxBits - 1 To 0 Step -1

    If (LowValue And 2 ^ X) = 0 Then

        Call AddBitsToOutStream(0, 1)

    Else

        Call AddBitsToOutStream(1, 1)

    End If

Next

Do While OutBitCount > 0

    Call AddBitsToOutStream(1, 1)

Loop

    ReDim ByteArray(OutPos - 1)

    Call CopyMem(ByteArray(0),
OutStream(0), OutPos)

    For OutBitCount = 0 To 8


    Next


End Sub


Public Sub
DeCompress_ArithMetic_DMC(ByteArray()
As Byte)

    Dim InpPos As Long

    Dim InBitPos As Integer

    Dim LowValue As Long

    Dim HighValue As Long

    Dim RangValue As Long

    Dim MidValue As Long
```

```
    Dim Value As Long
    Dim Char As Byte
    Dim X As Integer
    Dim Index As Integer
    Dim EOF_State As Boolean
    Dim TopBit As Long
    Dim One(256) As Long
    Dim Zero(256) As Long
    Call Init_Ari_Bit2
    LowValue = 0
    HighValue = (2 ^ MaxBits) - 1
    TopBit = 2 ^ (MaxBits - 1)
    InpPos = 0
    Value = ReadBitsFromArray(ByteArray,
InpPos, InBitPos, MaxBits)
    Index = -1
    For X = 0 To 256
       One(X) = 1
       Zero(X) = 1
    Next
    Do
       Char = 0
       For X = 0 To 7
          Index = (1 * (2 ^ X)) - 1 + Char
          RangValue = HighValue - LowValue
          MidValue = LowValue + (RangValue *
(Zero(Index) / (One(Index) + Zero(Index))))
          If MidValue = LowValue Then
MidValue = MidValue + 1
          If MidValue = HighValue - 1 Then
MidValue = MidValue - 1
          If Value >= MidValue Then
             Char = Char + Char + 1
             LowValue = MidValue
             One(Index) = One(Index) + 1
          Else
             Char = Char + Char
             HighValue = MidValue
             Zero(Index) = Zero(Index) + 1
          End If
          If AritmaticRescale = True Then
             If One(Index) > 127 Or Zero(Index)
> 127 Then
                One(Index) = Int(One(Index) / 2)
+ 1
                Zero(Index) = Int(Zero(Index) / 2)
+ 1
             End If
          End If
          Do While (HighValue And TopBit) =
(LowValue And TopBit) Or LowValue >
HighValue - 255
             If (LowValue And TopBit) = 1 Then
                Char = Char
             End If
             If InpPos <= UBound(ByteArray)
Then
                Value = (Value And (TopBit - 1))
* 2 + ReadBitsFromArray(ByteArray, InpPos,
InBitPos, 1)
                HighValue = (HighValue And
(TopBit - 1)) * 2 + 1
```

```
            LowValue = (LowValue And (TopBit - 1)) * 2

            If LowValue >= HighValue Then HighValue = (2 ^ MaxBits) - 1

          Else

            EOF_State = True

            Exit Do

          End If

        Loop

        If EOF_State = True Then Exit Do

      Next

      Call AddCharToArray(OutStream, OutPos, Char)

   Loop

   ReDim ByteArray(OutPos - 1)

   Call CopyMem(ByteArray(0), OutStream(0), OutPos)

End Sub



Private Sub Init_Ari_Bit2()

   Dim X As Integer

   ReDim OutStream(500)

   OutPos = 0

   OutBitCount = 0

   OutByteBuf = 0

End Sub


Private Sub AddBitsToOutStream(Number As Long, Numbits As Integer)

   Dim X As Long

   For X = Numbits - 1 To 0 Step -1

      OutByteBuf = OutByteBuf * 2 + (-1 * ((Number And CDbl(2 ^ X)) > 0))

      OutBitCount = OutBitCount + 1

      If OutBitCount = 8 Then

         OutStream(OutPos) = OutByteBuf

         OutBitCount = 0

         OutByteBuf = 0

         OutPos = OutPos + 1

         If OutPos > UBound(OutStream) Then

            ReDim Preserve OutStream(OutPos + 500)

         End If

      End If

   Next

End Sub


'this function will return a value out of the amaunt of bits you asked for

Private Function ReadBitsFromArray(FromArray() As Byte, FromPos As Long, FromBit As Integer, Numbits As Integer) As Long

   Dim X As Integer

   Dim Temp As Long

   For X = 1 To Numbits

      Temp = Temp * 2 + (-1 * ((FromArray(FromPos) And 2 ^ (7 - FromBit)) > 0))

      FromBit = FromBit + 1

      If FromBit = 8 Then
```

```
        If FromPos + 1 > UBound(FromArray) Then
            Do While X < Numbits
               Temp = Temp * 2
               X = X + 1
            Loop
            FromPos = FromPos + 1
            Exit For
         End If
         FromPos = FromPos + 1
         FromBit = 0
      End If
   Next
   ReadBitsFromArray = Temp
End Function


'this sub will add a char into the outputstream
Private Sub AddCharToArray(Toarray() As Byte, ToPos As Long, Char As Byte)
   If ToPos > UBound(Toarray) Then ReDim Preserve Toarray(ToPos + 500)
   Toarray(ToPos) = Char
   ToPos = ToPos + 1
End Sub
```

Prekompresi

Frequency Shifter

```
Option Explicit


Private Dictionary As String
Private CharCount(256) As Long


'This coder Makes Use of a dictionary of all ascii characters
'it will count the times a character is encountered
'Every time a certain character is encounterd it will be shifted
'forward in the directory untill it is in front or untill the character
'before it has a higher rate


Public Sub FrequentShifter_Coder(ByteArray() As Byte)
   Dim X As Long
   Dim Temp As Byte
   Call Init_FrequentShifter
   For X = 0 To UBound(ByteArray)
      Temp = ByteArray(X)
      ByteArray(X) = InStr(Dictionary, Chr(Temp)) - 1
      Call update_Model(Temp)
   Next
End Sub


Public Sub FrequentShifter_DeCoder(ByteArray() As Byte)
   Dim X, XX As Long
   Dim Temp As Byte
```

```
    Call Init_FrequentShifter

    For X = 0 To UBound(ByteArray)

        Temp = ASC(Mid(Dictionary, ByteArray(X)
+ 1, 1))

        ByteArray(X) = Temp

        Call update_Model(Temp)

    Next

End Sub


Private Sub Init_FrequentShifter()

    Dim X As Integer

    Dictionary = ""

    For X = 0 To 255

        Dictionary = Dictionary & Chr(X)

        CharCount(X) = 0

     Next

    CharCount(256) = 0

End Sub


Private Sub update_Model(Char As Byte)

    Dim Dictpos As Integer

    Dim OldPos As Integer

    Dim Temp As Long

    Dictpos = InStr(Dictionary, Chr(Char)) - 1

    OldPos = Dictpos

    CharCount(Dictpos) = CharCount(Dictpos) +
1

    Do While Dictpos > 0

        If CharCount(Dictpos) <
CharCount(Dictpos - 1) Then Exit Do

        Temp = CharCount(Dictpos - 1)

        CharCount(Dictpos - 1) =
CharCount(Dictpos)

        CharCount(Dictpos) = Temp

        Dictpos = Dictpos - 1

    Loop

    If OldPos = Dictpos Then Exit Sub

    Dictionary = Left(Dictionary, Dictpos) &
Chr(Char) & Mid(Dictionary, Dictpos + 1,
OldPos - Dictpos) & Mid(Dictionary, OldPos +
2)

End Sub
```

Burrows-Wheeler Transform

```
'This is a Burrows-Wheeler transform coder

'It works by sorting al the data in
lexicographical order

'and the it takes the last character of each array

'----------------------Transform----------------------
-

'The array must be seen as a circle so
LAST+1=FIRST

'then you make copies of it len(text) times but
every copy has shift 1 to the right

'then you can sort the strings

'

'example: Hariyanto

'

'   original                    sorted
```

```
'

'  0 = H a r i y a n t o      0 = 0 = H a r i y a n t
o

'  1 = a r i y a n t o H      1 = 5 = a n t o H a r i
y

'  2 = r i y a n t o H a      2 = 1 = a r i y a n t o
H

'  3 = i y a n t o H a r      3 = 3 = i y a n t o H a
r

'  4 = y a n t o H a r i      4 = 6 = n t o H a r i y
a

'  5 = a n t o H a r i y      5 = 8 = o H a r i y a n
t

'  6 = n t o H a r i y a      6 = 2 = r i y a n t o H
a

'  7 = t o H a r i y a n      7 = 7 = t o H a r i y a
n

'  8 = o H a r i y a n t      8 = 4 = y a n t o H a r
i

'

'if u take the last characters of the sorted strings
u'll get

'  oyHratani with prefix 2

'don't forget the prefix (without it you won't get
the original text back)

'----------Decode----------

'The thing we need to do is create another
string with the same contents

'and sort that other string so that we get

'

'place: 0 1 2 3 4 5 6 7 8

'org:   o y H r a t a n i

'new:   H a a i n o r t y

'

'now where gone create a transformation table

'If we take the first 'H' as position 0 and look it
up in the org we'll see

'that we find the first 'H' in place 2. This means
that TV(0)=2

'The first 'a' in new we'll find as the first 'a' in
org. so TV(1)=4

'after doing all the characters u will get a table
like this

'( 2 , 4 , 6 , 8 , 7 , 0 , 3 , 5 , 1 ) this is base 0

'with help if the prefix we now gen get the
original string back according to

'the next

'

'  Offset=prefix

'  For i = 0 To lenght of text

'      BWT_DeCodecString =
BWT_DeCodecString & Chr(L(Offset))

'      Offset = TV(Offset)

'  Next

'


Public Sub BWT_CodecArray4(ByteArray()
As Byte)

    Dim F() As Long

    Dim FTemp() As Long

    Dim OutStream() As Byte

    Dim i As Long

    Dim J As Long

    Dim b As Long
```

```
Dim L As Long
Dim t As Long
Dim r As Long
Dim d As Long
Dim K As Long
Dim Y As Long
Dim Z As Long
Dim Q As Integer
Dim ASC As Integer
Dim FileLength As Long
Dim P(1 To 100) As Long
Dim W(1 To 100) As Long
Dim X As Long
Dim Prefix As Long
Dim CharCount(255) As Long
Dim TempCount() As Long
Dim Spos(255) As Long
Dim TPos() As Long
    Dim CheckPos As Long
    Dim NuPos As Long
    FileLength = UBound(ByteArray)
    ReDim F(FileLength)
'This is the speedsort method wich is the fastest as far as i know
'first whe collect the frequentie of each char
    For X = 0 To FileLength
        CharCount(ByteArray(X)) = CharCount(ByteArray(X)) + 1
    Next
'then where gone create the offset pointers
    NuPos = 0
    For X = 0 To 255
        If CharCount(X) > 0 Then
            Spos(X) = NuPos
            NuPos = NuPos + CharCount(X)
        End If
    Next
'and last where place the pointers in order
    For X = 0 To FileLength
        F(Spos(ByteArray(X))) = X
        Spos(ByteArray(X)) = Spos(ByteArray(X)) + 1
    Next
'The BytePointers are now sorted
'and now where cone try to create lexicograpical sorted arrays
'Lets start with a speedsort method and finish the job with Quicksort
    For ASC = 0 To 255
        If CharCount(ASC) > 1 Then
            ReDim TempCount(255)
            ReDim TPos(255)
            ReDim FTemp(CharCount(ASC) - 1)
            NuPos = Spos(ASC) - CharCount(ASC)
            Z = 0
```

```
        For X = NuPos To NuPos +
CharCount(ASC) - 1

           FTemp(Z) = F(X)

           Z = Z + 1

        Next

        For X = 0 To CharCount(ASC) - 1

           Z = FTemp(X) + 1: If Z > FileLength
Then Z = Z - FileLength - 1

           TempCount(ByteArray(Z)) =
TempCount(ByteArray(Z)) + 1

        Next

        For X = 0 To 255

           If TempCount(X) > 0 Then

              TPos(X) = NuPos

              NuPos = NuPos + TempCount(X)

           End If

        Next

        For X = 0 To CharCount(ASC) - 1

           Z = FTemp(X) + 1: If Z > FileLength
Then Z = Z - FileLength - 1

           F(TPos(ByteArray(Z))) = FTemp(X)

           TPos(ByteArray(Z)) =
TPos(ByteArray(Z)) + 1

        Next

        NuPos = Spos(ASC) -
CharCount(ASC)

        For Q = 0 To 255

           If TempCount(Q) > 0 Then

              L = NuPos

              r = NuPos + TempCount(Q) - 1

              NuPos = NuPos + TempCount(Q)

              If TempCount(Q) > 1 Then GoSub
QuickSort

           End If

        Next

     End If

  Next

' The array is sorted so let get the last
characters and store them

' in the output stream

  ReDim OutStream(FileLength + 2)

  For i = 0 To FileLength

     If F(i) = 1 Then Prefix = i

     If F(i) = 0 Then

        OutStream(i) = ByteArray(FileLength)

     Else

        OutStream(i) = ByteArray(F(i) - 1)

     End If

  Next

  OutStream(FileLength + 1) = Int(Prefix /
&H100) And &HFF

  OutStream(FileLength + 2) = Prefix And
&HFF

end_Test:

  ReDim ByteArray(UBound(OutStream))

  Call CopyMem(ByteArray(0),
OutStream(0), UBound(OutStream) + 1)

  Exit Sub


QuickSort:
```

```
    K = 1
    P(K) = L
    W(K) = r
    d = 1
    Do
toploop:
        If r - L < 10 Then GoTo bubsort
        i = L
        J = r
        While J > i
            Y = F(i) + 1: If Y > FileLength Then Y
= Y - FileLength - 1
            Z = F(J) + 1: If Z > FileLength Then Z
= Z - FileLength - 1
            Do While ByteArray(Y) =
ByteArray(Z)
                Y = Y + 1: If Y > FileLength Then Y
= Y - FileLength - 1
                Z = Z + 1: If Z > FileLength Then Z
= Z - FileLength - 1
            Loop
            If ByteArray(Y) > ByteArray(Z) Then
                t = F(J)
                F(J) = F(i)
                F(i) = t
                d = -d
            End If
            If d = -1 Then
                J = J - 1
            Else
                i = i + 1
            End If
        Wend
        J = J + 1
        K = K + 1
        If i - L < r - J Then
            P(K) = J
            W(K) = r
            r = i
        Else
            P(K) = L
            W(K) = i
            L = J
        End If
        d = -d
        GoTo toploop
bubsort:
    If r - L > 0 Then
        For i = L To r
            b = i
            For J = b + 1 To r
                Y = F(J) + 1: If Y > FileLength Then
Y = Y - FileLength - 1
                Z = F(b) + 1: If Z > FileLength Then
Z = Z - FileLength - 1
                Do While ByteArray(Y) =
ByteArray(Z)
                    Y = Y + 1: If Y > FileLength Then
Y = Y - FileLength - 1
```

```
            Z = Z + 1: If Z > FileLength Then
Z = Z - FileLength - 1

            Loop

            If ByteArray(Y) < ByteArray(Z)
Then b = J

         Next J

         If i <> b Then

            t = F(b)

            F(b) = F(i)

            F(i) = t

         End If

      Next i

   End If

   L = P(K)

   r = W(K)

   K = K - 1

   Loop Until K = 0

   Return

End Sub


'Here whe gone restore the BWT-coded string

Public Sub
BWT_DeCodecArray4(ByteArray() As Byte)

   Dim TV() As Long

   Dim Spos(255) As Long

   Dim FileLength As Long

   Dim OffSet As Long

   Dim X As Long

   Dim Y As Long

   Dim NuPos As Long

   Dim CharCount(255) As Long

   Dim OutStream() As Byte

   FileLength = UBound(ByteArray)

'read the offset and restore the original size

   OffSet = CLng(ByteArray(FileLength - 1)) *
256 + ByteArray(FileLength)

   ReDim Preserve ByteArray(FileLength - 2)

   FileLength = UBound(ByteArray)

   ReDim OutStream(FileLength)

   ReDim TV(FileLength)

'Lets use the speedsort method to sort the array

'(no need to do it lexicographical)

   For X = 0 To FileLength

      CharCount(ByteArray(X)) =
CharCount(ByteArray(X)) + 1

   Next

   NuPos = 0

' Place the items in the sorted array.

   For X = 0 To 255

      Spos(X) = NuPos

      NuPos = NuPos + CharCount(X)

   Next

'Now whe have the original and the sorted
array so whe can construct

'a transformation tabel

   For X = 0 To FileLength
```

```
        TV(Spos(ByteArray(X))) = X

        Spos(ByteArray(X)) =
Spos(ByteArray(X)) + 1

    Next

'with use of the transformation tabel and the
offset whe can reconstruct

'the original data

    For X = 0 To FileLength

        OutStream(X) = ByteArray(OffSet)

        OffSet = TV(OffSet)

    Next

    Call CopyMem(ByteArray(0),
OutStream(0), UBound(OutStream) + 1)

End Sub
```

Move-to- Front

```
Option Explicit

'this is a Move To Front Coder which returns a
lot of

'small numbers because when a value is found
it will be

'placed at the start of the dictionary

'There are two methods in this module

'

'The first one uses a standard dictionary
excisting of all the

'ascii characters

'The second one creates a dictionary while it is
coding

'this dictionary has to be stored to get the
decoder work


Public Sub MTF_CoderArray(bytes() As Byte,
Optional Dictionary As String = "")

    Dim DictString As String

    Dim NewPos As Integer

    Dim X As Long

    If Dictionary = "" Then

        For X = 0 To 255

            DictString = DictString & Chr(X)

        Next

    Else

        DictString = Dictionary

    End If

    For X = 0 To UBound(bytes)

        NewPos = InStr(DictString,
Chr(bytes(X)))

        DictString = Chr(bytes(X)) &
Left(DictString, NewPos - 1) &
Mid(DictString, NewPos + 1)

        bytes(X) = NewPos - 1

    Next

End Sub


Public Sub MTF_DeCoderArray(bytes() As
Byte, Optional Dictionary As String = "")

    Dim DictString As String

    Dim NewString As String

    Dim NewPos As Integer

    Dim X As Long
```

```
    If Dictionary = "" Then
        For X = 0 To 255
            DictString = DictString & Chr(X)
        Next
    Else
        DictString = Dictionary
    End If
    For X = 0 To UBound(bytes)
        NewPos = bytes(X) + 1
        bytes(X) = ASC(Mid(DictString,
NewPos, 1))
        DictString = Mid(DictString, NewPos, 1)
& Left(DictString, NewPos - 1) &
Mid(DictString, NewPos + 1)
    Next
End Sub


Public Sub MTF_CoderArray2(ByteArray() As
Byte)
    Dim DictString As String
    Dim OrgDict As String
    Dim NewPos As Integer
    Dim X As Long
    Dim Dictpos As Long
    For X = 0 To UBound(ByteArray)
        If InStr(DictString, Chr(ByteArray(X))) =
0 Then DictString = DictString &
Chr(ByteArray(X)): OrgDict = OrgDict &
Chr(ByteArray(X))
        NewPos = InStr(DictString,
Chr(ByteArray(X)))
        DictString = Chr(ByteArray(X)) &
Left(DictString, NewPos - 1) &
Mid(DictString, NewPos + 1)
        ByteArray(X) = NewPos - 1
    Next
    Dictpos = UBound(ByteArray) + 1
    ReDim Preserve ByteArray(Len(OrgDict) +
1 + UBound(ByteArray))
    For X = 1 To Len(OrgDict)
        ByteArray(Dictpos) = ASC(Mid(OrgDict,
X, 1))
        Dictpos = Dictpos + 1
    Next
    ByteArray(Dictpos) = Len(OrgDict) - 1
End Sub


Public Sub MTF_DeCoderArray2(ByteArray()
As Byte)
    Dim DictString As String
    Dim DictLen As Integer
    Dim NewPos As Integer
    Dim X As Long
    DictLen = ByteArray(UBound(ByteArray))
+ 1
    For X = UBound(ByteArray) - DictLen To
UBound(ByteArray) - 1
        DictString = DictString &
Chr(ByteArray(X))
    Next
    ReDim Preserve
ByteArray(UBound(ByteArray) - DictLen - 1)
    For X = 0 To UBound(ByteArray)
```

```
        NewPos = ByteArray(X) + 1

        ByteArray(X) = ASC(Mid(DictString,
NewPos, 1))

        DictString = Mid(DictString, NewPos, 1)
& Left(DictString, NewPos - 1) &
Mid(DictString, NewPos + 1)

    Next

End Sub
```

Global module

```
Public Declare Sub CopyMem Lib
"kernel32" Alias "RtlMoveMemory"
(Destination As Any, source As Any,
ByVal Length As Long)


Public RGBColor(6) As ColorConstants
'color codes for graphics

Public OriginalArray() As Byte
'array to store the original

Public OriginalSize As Long          'size
of the original file

Public WorkArray() As Byte
'array to store the results

Public LoadFileName As String
'which file is loeded

Public JustLoaded As Boolean         'to
see if the original file is just loaded

Public DictionarySize As Integer     'for
use with LZW and LZSS compressors

Public LastCoder As Integer          'wich
coder is used last

Public UsedCodecs() As Integer       'to
store the codecs used

Public LastDeCoded As Boolean        'to
see what has happen lately

Public AritmaticRescale As Boolean   'to
set rescale on

Public CompDecomp As Integer
'result from compress or decompress
screen

Public ParingType As Byte            'Byte
used to determine the type of paring


'constants for the coder types

Public Const Coder_FrequentieShifter = 1

Public Const Coder_BWT = 2

Public Const Coder_MTF_No_Header = 3


Private CodName(3) As String


'constants for the compressor types

Public Const
Compressor_Arithmetic_DMC = 1


Public CompName(1) As String

Public Sub Copy_Orig2Work()

  ReDim
WorkArray(UBound(OriginalArray))

  Call CopyMem(WorkArray(0),
OriginalArray(0), UBound(OriginalArray) +
1)

End Sub
```

```
'copy the workarray to the original array so that whe can apply a

'second compress/coder on the file

Public Sub Copy_Work2Orig()

   ReDim OriginalArray(UBound(WorkArray))

   Call CopyMem(OriginalArray(0), WorkArray(0), UBound(WorkArray) + 1)

End Sub


'This sub is used to start a coder

'whichcoder is the constant of the codertype

'Decode is used to say if we want to code or decode

Public Sub Start_Coder(WichCoder)

   Dim Decode As Boolean

   Dim StTime As Double

   Dim Text As String

   Dim LastUsed As Integer

   If UBound(OriginalArray) = 0 Then

      MsgBox "There is nothing to code/Decode"

      Exit Sub

   End If

   If AutoDecodeIsOn = False Then

      Decode = True

      frmCodeDecode.Show vbModal

      DoEvents

      If CompDecomp = 0 Then Exit Sub

      If CompDecomp = 1 Then Decode = False

   Else

      Decode = True

   End If

   If Decode = True Then

      LastUsed = UsedCodecs(UBound(UsedCodecs))

      If JustLoaded = True Then LastUsed = 0

      If (WichCoder Or 128) <> LastUsed Then

         Text = "This is not coded with the " & CodName(WichCoder) & Chr(13)

         If LastUsed = 0 Then

            Text = Text & "Its not coded at all"

         Else

            If LastUsed > 128 Then

               Text = Text & "Its coded with the " & CodName(LastUsed And 127)

            Else

               Text = Text & "Its compressed with " & CompName(LastUsed)

            End If

         End If
```

```
        MsgBox Text
        Exit Sub
    End If
  Else
    LastCoder = WichCoder Or 128
  End If
  LastDeCoded = Decode
  Call Copy_Orig2Work
  If JustLoaded = True Then
    JustLoaded = False
    ReDim UsedCodecs(0)
  End If
  StTime = Timer
  master.MousePointer =
MousePointerConstants.vbHourglass
  Select Case WichCoder
      Case 1
        If Decode = False Then Call
FrequentShifter_Coder(WorkArray)
        If Decode = True Then Call
FrequentShifter_DeCoder(WorkArray)
      Case 2
        If Decode = False Then Call
BWT_CodecArray4(WorkArray)
        If Decode = True Then Call
BWT_DeCodecArray4(WorkArray)
      Case 3
        If Decode = False Then Call
MTF_CoderArray(WorkArray)
        If Decode = True Then Call
MTF_DeCoderArray(WorkArray)
  End Select
  master.MousePointer =
MousePointerConstants.vbDefault
  If AutoDecodeIsOn = False Then
      Call Show_Statistics(False,
WorkArray, Timer - StTime)
  End If
End Sub


'This sub is used to start a compressor
'whichcoder is the constant of the
compressortype
'Decode is used to say if we want to
compress or decompress
Public Sub
Start_Compressor(WichCompressor)
  Dim Decompress As Boolean
  Dim Dummy As Boolean
  Dim StTime As Double
  Dim Text As String
  Dim LastUsed As Integer
  If UBound(OriginalArray) = 0 Then
```

```
      MsgBox "There is nothing to
compress/Decompress"

      Exit Sub

   End If

   If AutoDecodeIsOn = False Then

      Decompress = True

      frmCompDecomp.Show vbModal

      DoEvents

      If CompDecomp = 0 Then Exit Sub

      If CompDecomp = 1 Then
Decompress = False

   Else

      Decompress = True

   End If

   If Decompress = True Then

      LastUsed =
UsedCodecs(UBound(UsedCodecs))

      If JustLoaded = True Then LastUsed =
0

      If WichCompressor <> LastUsed Then

         Text = "This is not compressed with
" & CompName(WichCompressor) & "." &
Chr(13)

         If LastUsed = 0 Then

            Text = Text & "Its not
compressed at all"

         Else

            If LastUsed > 128 Then

               Text = Text & "Its coded with
the " & CodName(LastUsed And 127)

            Else

               Text = Text & "Its compressed
with " & CompName(LastUsed)

            End If

         End If

         MsgBox Text

         Exit Sub

      End If

   Else

      LastCoder = WichCompressor

   End If

   Call Copy_Orig2Work

   LastDeCoded = Decompress

   If JustLoaded = True Then

      JustLoaded = False

      ReDim UsedCodecs(0)

   End If

   master.MousePointer =
MousePointerConstants.vbHourglass

   StTime = Timer

   Select Case WichCompressor

      Case 4

         If Decompress = False Then Call
Compress_ArithMetic_DMC(WorkArray)

         If Decompress = True Then Call
DeCompress_ArithMetic_DMC(WorkArray
)
```

```
    End Select

    master.MousePointer =
MousePointerConstants.vbDefault

    If AutoDecodeIsOn = False Then

        Call Show_Statistics(False,
WorkArray, Timer - StTime)

    End If

End Sub

'this sub is used to load a chosen file

Public Sub load_File(Name As String)

    Dim FreeNum As Integer

    If Name = "" Then Exit Sub

    FreeNum = FreeFile

    Open Name For Binary As #FreeNum

    ReDim OriginalArray(0 To
LOF(FreeNum) - 1)

    Get #FreeNum, , OriginalArray()

    Close #FreeNum

    JustLoaded = True

    Call
Split_Header_From_File(OriginalArray)

    master.Caption = "Test Programm For
Compressors  [file = " & LoadFileName &
"]"

    OriginalSize = UBound(OriginalArray) +
1

    Call Show_Statistics(True, OriginalArray)

End Sub

'this sub is used to see if the file just
loaded is a file which is

'stored by this programm and is already
coded/compressed

Private Sub
Split_Header_From_File(ByteArray() As
Byte)

    Dim HeadText As String

    Dim X As Integer

    Dim CodecsUsed As Integer

    Dim Version As String

    Dim InPos As Long

    If UBound(ByteArray) < 3 Then Exit Sub
'original file to small

    InPos = UBound(ByteArray)

    For X = 0 To 2

        HeadText = HeadText &
Chr(ByteArray(InPos))

        InPos = InPos - 1

    Next

    If HeadText <> "UCF" Then Exit Sub
'this is an un-UCF'ed file

    Version = Chr(ByteArray(InPos))

    InPos = InPos - 1

    Select Case Version

        Case "0"

            CodecsUsed = ByteArray(InPos)

            InPos = InPos - 1
```

```
        ReDim UsedCodecs(CodecsUsed)
        For X = 1 To CodecsUsed
          UsedCodecs(X) = ByteArray(InPos)
          InPos = InPos - 1
        Next
        ReDim Preserve ByteArray(InPos)
    End Select
    ReDim WorkArray(0)
    For X = 0 To 255
      master.Bars(1 * 256 + X).Visible = False
    Next
    master.AscTab(1).Clear
    master.FreqTab(1).Clear
    master.FileSize(1).Caption = " "
    master.MaxValue(1).Caption = "Maximum"
    master.MidValue(1).Caption = "Medium"
    master.LowValue(1).Caption = "Lowest"
    JustLoaded = False
End Sub


'this sub is used to save a file
'if the file was coded/compressed the types of coders/compressors used
'will be saved with the file so that we can recall it later
Public Sub Save_File_As(ByteArray() As Byte, source As Boolean)
    Dim FileNr As Integer
    Dim HeadArray() As Byte
    Dim OutHead As Integer
    Dim HeadText As String
    Dim Answer As Integer
    Dim CodecsUsed As Integer
    Dim SaveName As String
    Dim ExtPos As Integer
    Dim Temp As Integer
    Dim X As Integer
    If UBound(ByteArray) = 0 Then
      MsgBox "There is nothing to be saved"
      Exit Sub
    End If
    If source = False And LastCoder <> 0 Then Call AddCoder2List(LastCoder)
    If UBound(UsedCodecs) = 0 And UBound(ByteArray) = UBound(OriginalArray) Then
      Answer = MsgBox("The file to save is the same as the original file" & Chr(13) & "Still want to save this file", vbYesNo + vbExclamation)
      If Answer = vbNo Then
        Exit Sub
```

```
    End If
  End If
Ask_SaveName:
  SaveName = ""
  master.Cdlg.DialogTitle = "Type in the
name you want to save with"
  master.Cdlg.FileName = ""
  master.Cdlg.ShowSave
  SaveName = master.Cdlg.FileName
  If SaveName = "" Then
    If source = False And LastCoder <> 0
Then
      ReDim Preserve
UsedCodecs(UBound(UsedCodecs) - 1)
      LastCoder =
UsedCodecs(UBound(UsedCodecs))
    End If
    Exit Sub
  End If
  Temp = 0
  Do
    ExtPos = Temp
    Temp = InStr(ExtPos + 1, SaveName,
".")
  Loop While Temp <> 0
  If ExtPos = 0 Or ExtPos < Len(SaveName)
- 5 Then
    SaveName = SaveName & ".xxx"
  End If
'store the header in reversed order at the
end of the file
  HeadText = "UCF0"
  If LastCoder = 0 And source = False Then
    CodecsUsed = 0
  Else
    CodecsUsed = UBound(UsedCodecs)
  End If
  ReDim HeadArray(4 + CodecsUsed)
  OutHead = 0
  For X = CodecsUsed To 1 Step -1
    HeadArray(OutHead) =
UsedCodecs(X)
    OutHead = OutHead + 1
  Next
  HeadArray(OutHead) = CodecsUsed
  OutHead = OutHead + 1
  For X = Len(HeadText) To 1 Step -1
    HeadArray(OutHead) =
ASC(Mid(HeadText, X, 1))
    OutHead = OutHead + 1
  Next
  FileNr = FreeFile
  If Dir(SaveName, vbNormal) <> "" Then
    Answer = MsgBox("File already exists"
& Chr(13) & Chr(13) & "Overwrite",
vbCritical + vbYesNo)
```

```
      If Answer = vbNo Then

         GoTo Ask_SaveName

      End If

      Kill SaveName   'first remove it
otherwise size is not adjusted

   End If

   Open SaveName For Binary As #FileNr

   Put #FileNr, , ByteArray()

   If CodecsUsed > 0 Then

      Put #FileNr, , HeadArray()

   End If

   Close #FileNr

End Sub


'this sub is used to show the statistics of a
file

'it can display both the original as the
workarray

Public Sub Show_Statistics(OrgData As
Boolean, Data() As Byte, Optional
TimeUsed As Double = 0)

   Dim StatWindow As Integer

   Dim Frequentie(255) As Long

   Dim SortFreq(1, 255) As Long

   Dim Counts() As Long

   Dim X As Long

   Dim Minval As Long

   Dim Maxval As Long

   Dim next_offset As Long

   Dim this_count As Long

   Dim HeightValue As Double

   Dim Entry As String

   Dim NewSize As String

   Dim NuSize As Long

   Dim BPB As String

   If OrgData = False Then StatWindow = 1

   NuSize = UBound(Data) + 1

   BPB = Format(((NuSize * 8) /
OriginalSize), "###0.000") & " bpb"

   NewSize = NuSize & " Bytes  [ " &
Format(100 - (OriginalSize - NuSize) /
OriginalSize * 100, "##0.00") & "% ]  "

   If TimeUsed > 0 Then

      NewSize = NewSize & BPB & "  " &
Format(TimeUsed, "###0.00") & " Sec."

   End If

   For X = 0 To UBound(Data)

      Frequentie(Data(X)) =
Frequentie(Data(X)) + 1

   Next

   Minval = UBound(Data)

   For X = 0 To 255

      If Minval > Frequentie(X) Then Minval
= Frequentie(X)

      If Maxval < Frequentie(X) Then
Maxval = Frequentie(X)
```

```
   Next

' Lets use the counting sort to sort them into another array

' Create the Counts array.

   ReDim Counts(Minval To Maxval)

' Count the items.

   For X = 0 To 255

      Counts(Frequentie(X)) = Counts(Frequentie(X)) + 1

   Next X

' Convert the counts into offsets.

   next_offset = 0

   For X = Maxval To Minval Step -1

      this_count = Counts(X)

      Counts(X) = next_offset

      next_offset = next_offset + this_count

   Next X

' Place the items in the sorted array.

   For X = 0 To 255

      SortFreq(0, Counts(Frequentie(X))) = Frequentie(X)

      SortFreq(1, Counts(Frequentie(X))) = X

      Counts(Frequentie(X)) = Counts(Frequentie(X)) + 1

   Next X

'Create the graphics view

   HeightValue = (SortFreq(0, 0) - SortFreq(0, 255)) / master.Graphic(StatWindow).Height

   For X = 0 To 255

      If Frequentie(X) - SortFreq(0, 255) <> 0 Then

         master.Bars(StatWindow * 256 + X).Visible = True

         master.Bars(StatWindow * 256 + X).Y1 = master.Bars(StatWindow * 256 + X).Y2 - (Frequentie(X) - SortFreq(0, 255)) / HeightValue

         master.Bars(StatWindow * 256 + X).BorderColor = RGBColor(X Mod 7)

      Else

         master.Bars(StatWindow * 256 + X).Visible = False

      End If

   Next

   master.MaxValue(StatWindow).Caption = SortFreq(0, 0)

   master.MidValue(StatWindow).Caption = Int((SortFreq(0, 0) - SortFreq(0, 255)) / 2) + SortFreq(0, 255)

   master.LowValue(StatWindow) = SortFreq(0, 255)

   master.FileSize(StatWindow).Caption = NewSize

'Create the statistics view

   master.AscTab(StatWindow).Clear
```

```
    master.FreqTab(StatWindow).Clear
    Entry = "Index    Freq."
    master.AscTab(StatWindow).AddItem Entry
    For X = 0 To 255
        Entry = Format(X, "##0") & Chr(9) & Frequentie(X)
        master.AscTab(StatWindow).AddItem Entry
    Next
    Entry = "Index" & Chr(9) & "Ascii" & Chr(9) & "Freq."
    master.FreqTab(StatWindow).AddItem Entry
    For X = 0 To 255
        Entry = Format(X, "##0") & Chr(9) & SortFreq(1, X) & Chr(9) & SortFreq(0, X)

master.FreqTab(StatWindow).AddItem Entry
    Next
End Sub


'this sub is used to show the contents of the file
'its used for both the original as the workarray
Public Sub Show_Contents(ByteArray() As Byte)
    Dim X As Long
    Dim Y As Integer
    Dim AddData As String
    Dim AddText As String
    Dim Data As Byte
    Dim Text As String
    On Error GoTo No_Data
    X = UBound(ByteArray)
    If X = 0 Then
        MsgBox "Ther is nothing to see because there is no data"
        Exit Sub
    End If
    On Error GoTo 0
    frmViewContents.Show
    frmViewContents.lstContents.Clear
    For X = 0 To UBound(ByteArray) Step 16
        AddData = String(61, " ")
        Mid(AddData, 35, 1) = "|"
        AddText = String(16, " ")
        Mid(AddData, 1, 9) = Right("00000000" & Hex(X), 8) & ":"
        For Y = 0 To 15
            If X + Y <= UBound(ByteArray) Then
                Data = ByteArray(X + Y)
                Mid(AddData, 12 + (3 * Y), 2) = Right("0" & Hex(Data), 2)
```

```
            If Data < 28 Then Text = Chr(1)
Else Text = Chr(Data)

            Mid(AddText, Y + 1, 1) = Text

         End If

      Next

      If frmViewContents.Visible = True
Then


frmViewContents.lstContents.AddItem
AddData & AddText

      Else

         Exit Sub

      End If

      If X Mod 500 * 16 = 0 Then DoEvents

   Next

   DoEvents

No_Data:

End Sub
```

```
'this sub is used to store a
coder/compressor type into an array

'so that we can keep up which
coders/compressors are used to get to

'the last file whe have standing in the
original array

Public Sub AddCoder2List(CodeNumber As
Integer)

   JustLoaded = False

   If LastDeCoded = True Then

      If UBound(UsedCodecs) > 0 Then

         ReDim Preserve
UsedCodecs(UBound(UsedCodecs) - 1)

         LastCoder =
UsedCodecs(UBound(UsedCodecs))

      End If

      Exit Sub

   End If

   ReDim Preserve
UsedCodecs(UBound(UsedCodecs) + 1)
   UsedCodecs(UBound(UsedCodecs)) =
CodeNumber

End Sub


'this sub is used to decode/uncompress
automaticly without the user

'having search which type of
coder/compressor was used

Public Sub Auto_Decode_Depack()

   Dim X As Integer

   Dim CodeNumber As Integer

   If UBound(OriginalArray) = 0 Then

      MsgBox "There is nothing to
Decode/Decompress"

      Exit Sub

   End If

   If UBound(UsedCodecs) = 0 Or
JustLoaded = True Then
```

```
      MsgBox "This file was'nt
Coded/Compressed"

      Exit Sub

   End If

   CodeNumber =
UsedCodecs(UBound(UsedCodecs))

   AutoDecodeIsOn = True

   If CodeNumber > 128 Then

      Call Start_Coder(CodeNumber And
127)

   Else

      Call Start_Compressor(CodeNumber)

   End If

   Call Copy_Work2Orig

   ReDim WorkArray(0)

   For X = 0 To 255

      master.Bars(1 * 256 + X).Visible =
False

   Next

   master.AscTab(1).Clear

   master.FreqTab(1).Clear

   master.FileSize(1).Caption = " "

   master.MaxValue(1).Caption =
"Maximum"

   master.MidValue(1).Caption =
"Medium"

   master.LowValue(1).Caption = "Lowest"

   AutoDecodeIsOn = False

   If UBound(UsedCodecs) > 0 Then

      ReDim Preserve
UsedCodecs(UBound(UsedCodecs) - 1)

      LastCoder =
UsedCodecs(UBound(UsedCodecs))

   End If

   Call Show_Statistics(True, OriginalArray)

End Sub
```

```
'this sub is used to compare the original
array with the workarray

Public Sub
Compare_Source_With_Target()

   Dim FileSize As Long

   Dim SameSize As Boolean

   Dim Text As String

   Dim Equal As Boolean

   Dim X As Long

   SameSize = True

   Equal = True

   If UBound(OriginalArray) = 0 Then

      MsgBox "There is nothing to
compare"

      Exit Sub

   End If

   If UBound(WorkArray) = 0 Then

      MsgBox "There is nothing to compare
with"
```

```
    Exit Sub
  End If
  FileSize = UBound(OriginalArray)
  If UBound(WorkArray) <> FileSize Then
    SameSize = False
    If UBound(WorkArray) < FileSize Then
      FileSize = UBound(WorkArray)
    End If
  End If
  For X = 0 To FileSize
    If OriginalArray(X) <> WorkArray(X)
Then
      Equal = False
      Exit For
    End If
  Next
  If Equal = False Then
    Text = "The files are different at
position " & X
    If SameSize = False Then
      Text = Text & Chr(13) & "And They
dont have the same size"
    End If
    MsgBox Text
    Exit Sub
  End If
  If SameSize = False Then
    Text = "The files are almost the same
except that they don't have the same size"
  Else
    Text = "the two files are the same"
  End If
  MsgBox Text
End Sub
Private Sub AriDMC_Click()
```

Master Form

```
  Call
Start_Compressor(Compressor_Arithmeti
c_DMC)
End Sub
Private Sub btnViewContentsOrig_Click()
  Call Show_Contents(OriginalArray)
End Sub

Private Sub
btnViewContentsTarget_Click()
  Call Show_Contents(WorkArray)
End Sub

Private Sub BWT_Click()
  Call Start_Coder(Coder_BWT)
End Sub
```

```
Private Sub CalcCRC32Source_Click()
   Dim CRCSource As Long
   On Error GoTo No_Data
   CRCSource = UBound(OriginalArray)
   On Error GoTo 0
   CRCSource = calcCRC32(OriginalArray)
   MsgBox Hex(CRCSource)
No_Data:
End Sub

Private Sub CalcCRC32Target_Click()
   Dim CRCTarget As Long
   On Error GoTo No_Data
   CRCTarget = UBound(WorkArray)
   On Error GoTo 0
   CRCTarget = calcCRC32(WorkArray)
   MsgBox Hex(CRCTarget)
No_Data:
End Sub

Private Sub CalculateEntropy_Click()
   Call Calculate_Entropy(OriginalArray)
End Sub

Private Sub CompareSWT_Click()
   Call Compare_Source_With_Target
End Sub

Private Sub CopyWorkToOrg_Click()
   If UBound(WorkArray) = 0 Then
      MsgBox "There is nothing to copy"
      Exit Sub
   End If
   Call Copy_Work2Orig
   Call AddCoder2List(LastCoder)
   Call Show_Statistics(True, OriginalArray)
End Sub

Private Sub ExitProg_Click()
   End
End Sub

Private Sub Form_Load()
   Dim X As Integer
   Dim Y As Integer
   Dim MaxWidth As Double
   For X = 0 To 1
      MaxWidth = Graphic(X).Width / 256
      For Y = 0 To 255
         Bars(X * 256 + Y).BorderWidth = 1
```

```
        Bars(X * 256 + Y).BorderStyle = 1

        Bars(X * 256 + Y).X1 =
Graphic(X).Left + Y * MaxWidth

        Bars(X * 256 + Y).X2 = Bars(X * 256
+ Y).X1

        Bars(X * 256 + Y).Y2 =
Graphic(X).Top + Graphic(X).Height

        Bars(X * 256 + Y).Y1 = Bars(X * 256
+ Y).Y2

        Bars(X * 256 + Y).Visible = True

      Next

   Next

   RGBColor(0) = vbBlue

   RGBColor(1) = vbCyan

   RGBColor(2) = vbGreen

   RGBColor(3) = vbMagenta

   RGBColor(4) = vbRed

   RGBColor(5) = vbYellow

   RGBColor(6) = vbWhite

   Call Init_CoderNameDataBase

   ReDim OriginalArray(0)

   ReDim WorkArray(0)

   ReDim UsedCodecs(0)

End Sub


Private Sub Form_Unload(Cancel As
Integer)

   End

End Sub


Private Sub FreqShift_Click()

   Call
Start_Coder(Coder_FrequentieShifter)

End Sub


Private Sub Load_Click()

   Dim OldFileName As String

   OldFileName = LoadFileName

   Cdlg.DialogTitle = "Select the file you
want to explore"

   Cdlg.FileName = ""

   Cdlg.ShowOpen

   LoadFileName = Cdlg.FileName

   Call load_File(LoadFileName)

   If LoadFileName = "" Then
LoadFileName = OldFileName

End Sub


Private Sub MTF_Click()

   Call
Start_Coder(Coder_MTF_No_Header)

End Sub


Private Sub RestoreOrig_Click()

   If LoadFileName = "" Then
```

```
        MsgBox "There was'nt yet original
data"

        Exit Sub

    End If

    Call load_File(LoadFileName)

End Sub
```

Private Sub Save_Click()

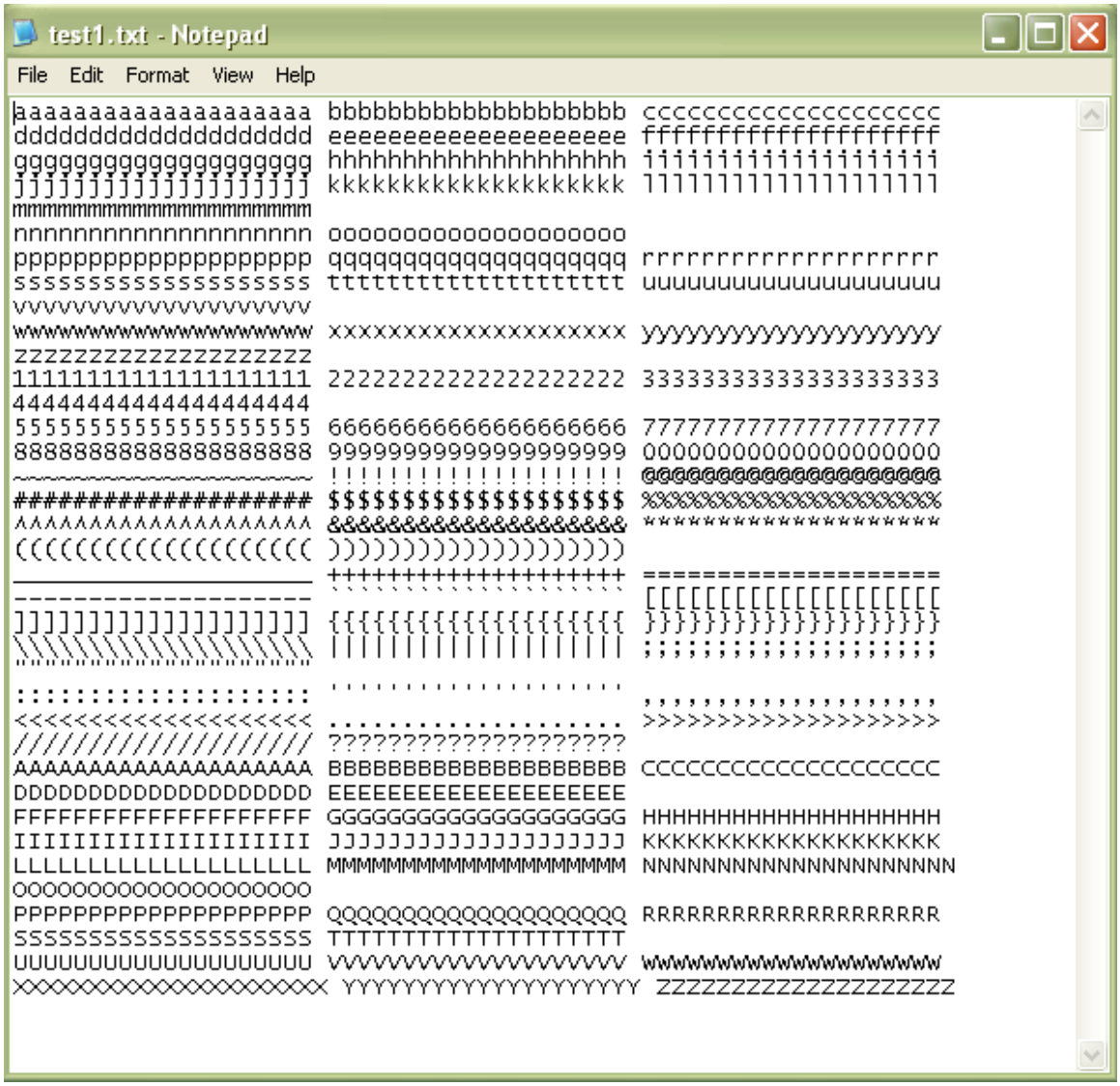
```
    Call Save_File_As(WorkArray, False)

End Sub
```

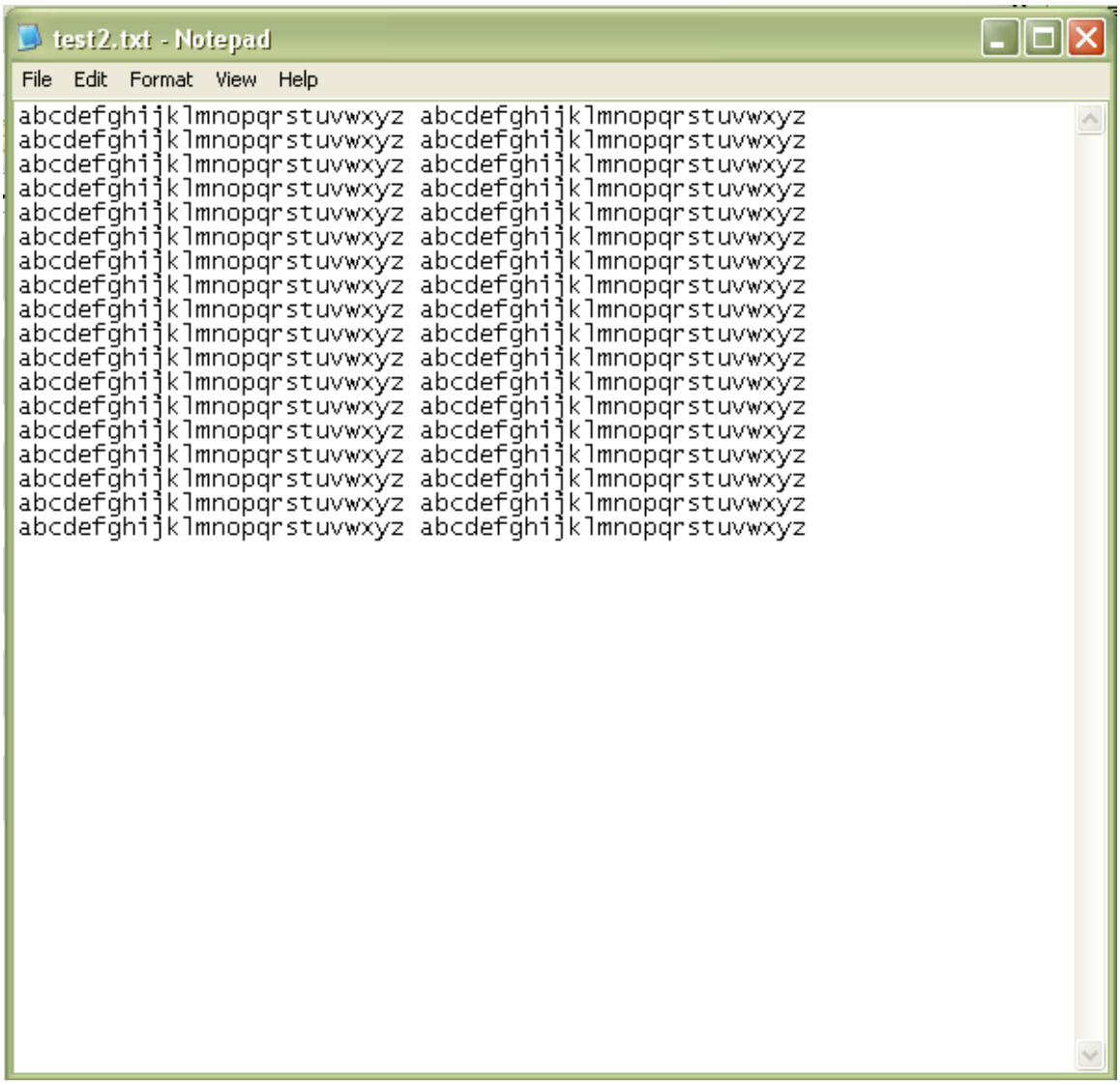
LAMPIRAN B

CUPLIKAN ISI FILE ASLI

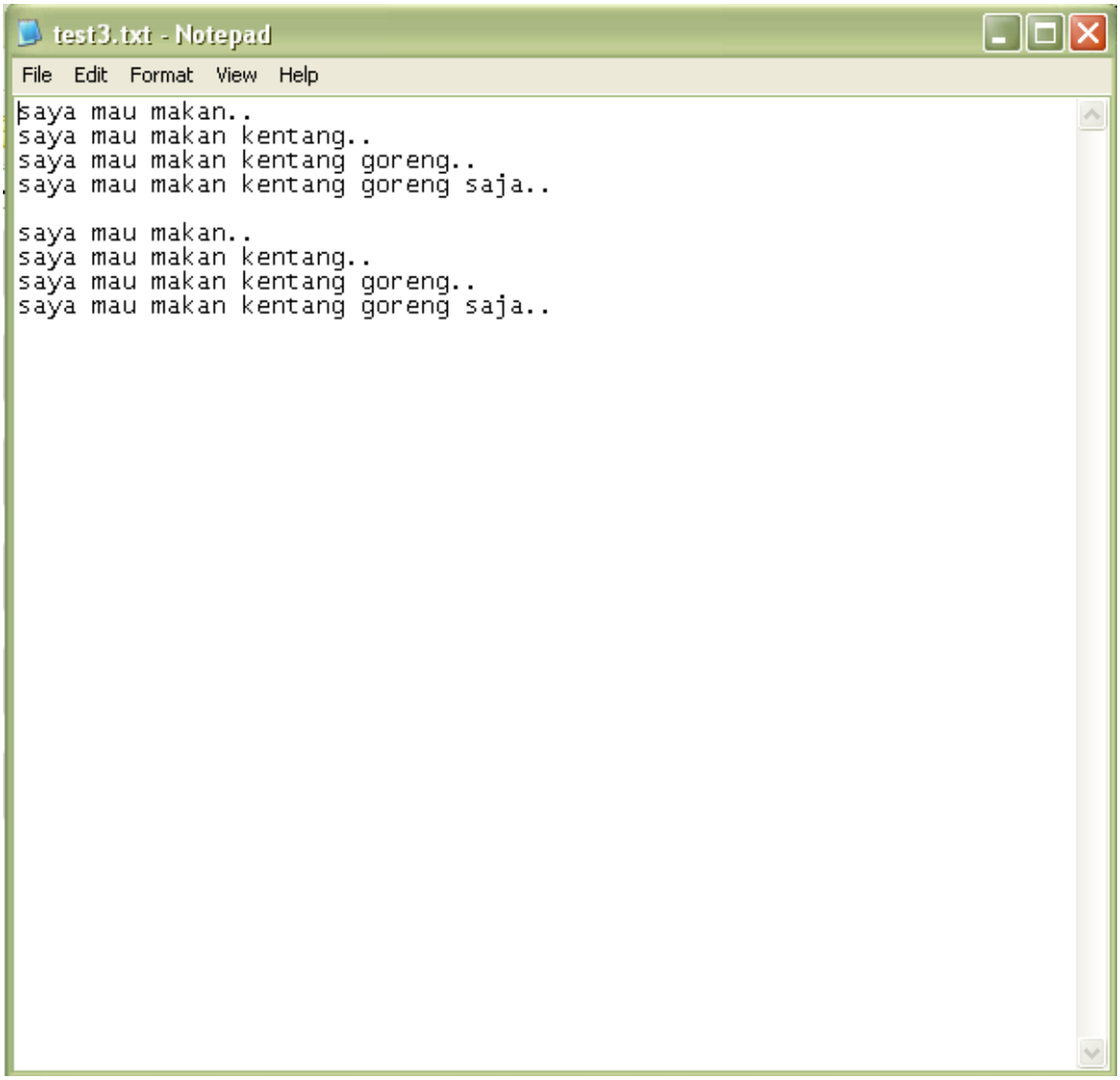
1. **test1.txt**



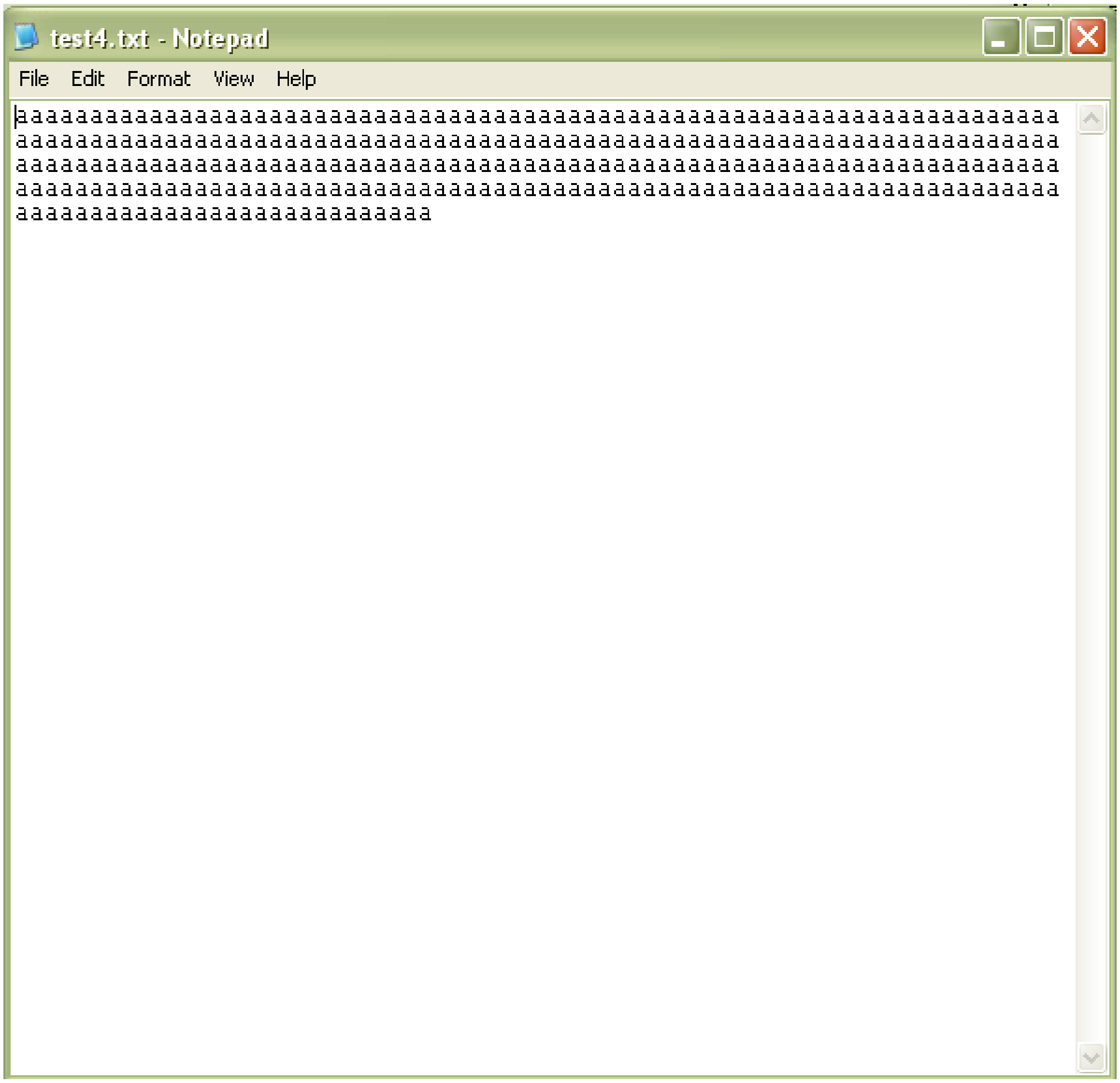
2. test2.txt


test2.txt - Notepad
File Edit Format View Help
abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvwxyz

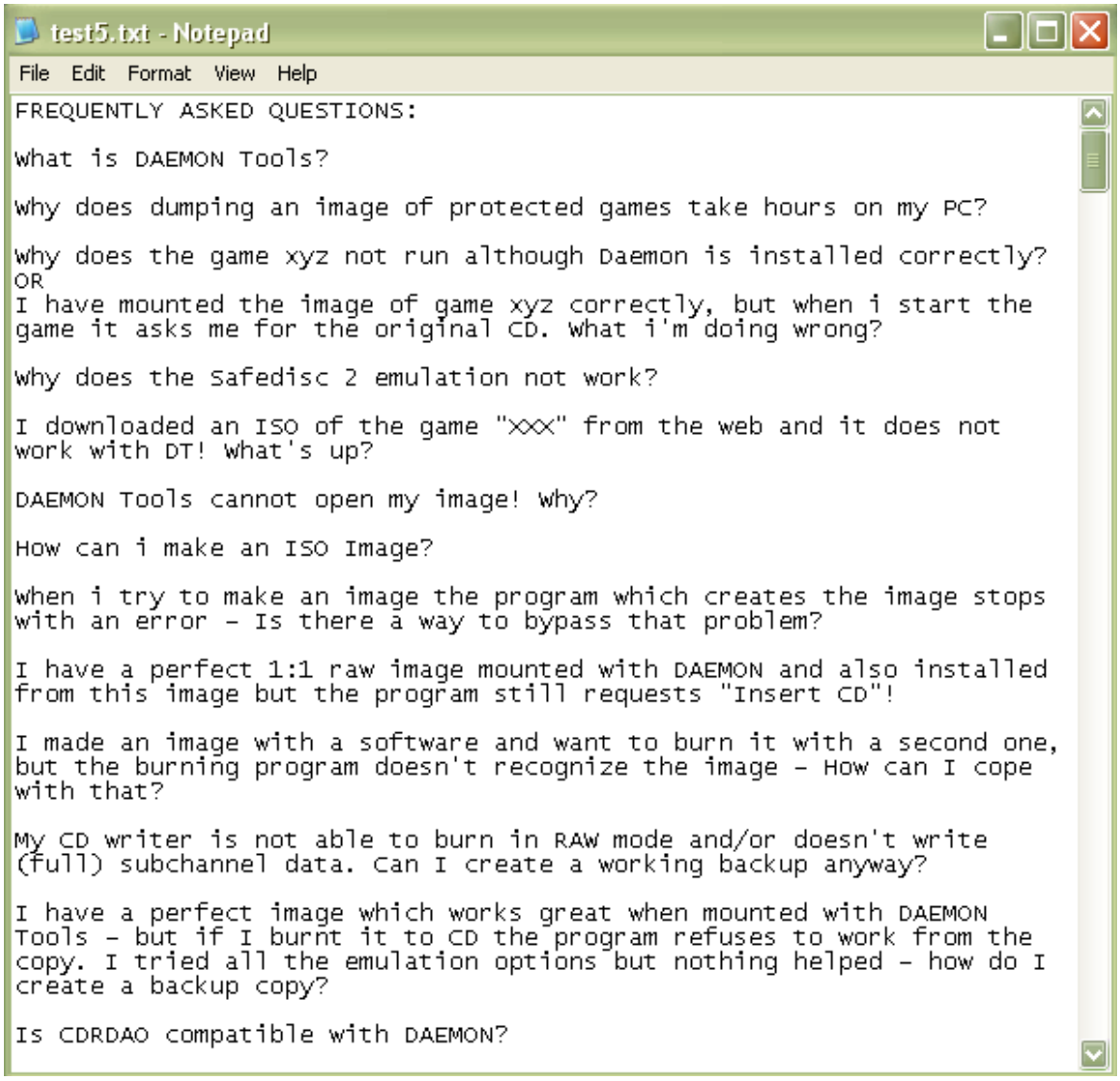
3. test3.txt

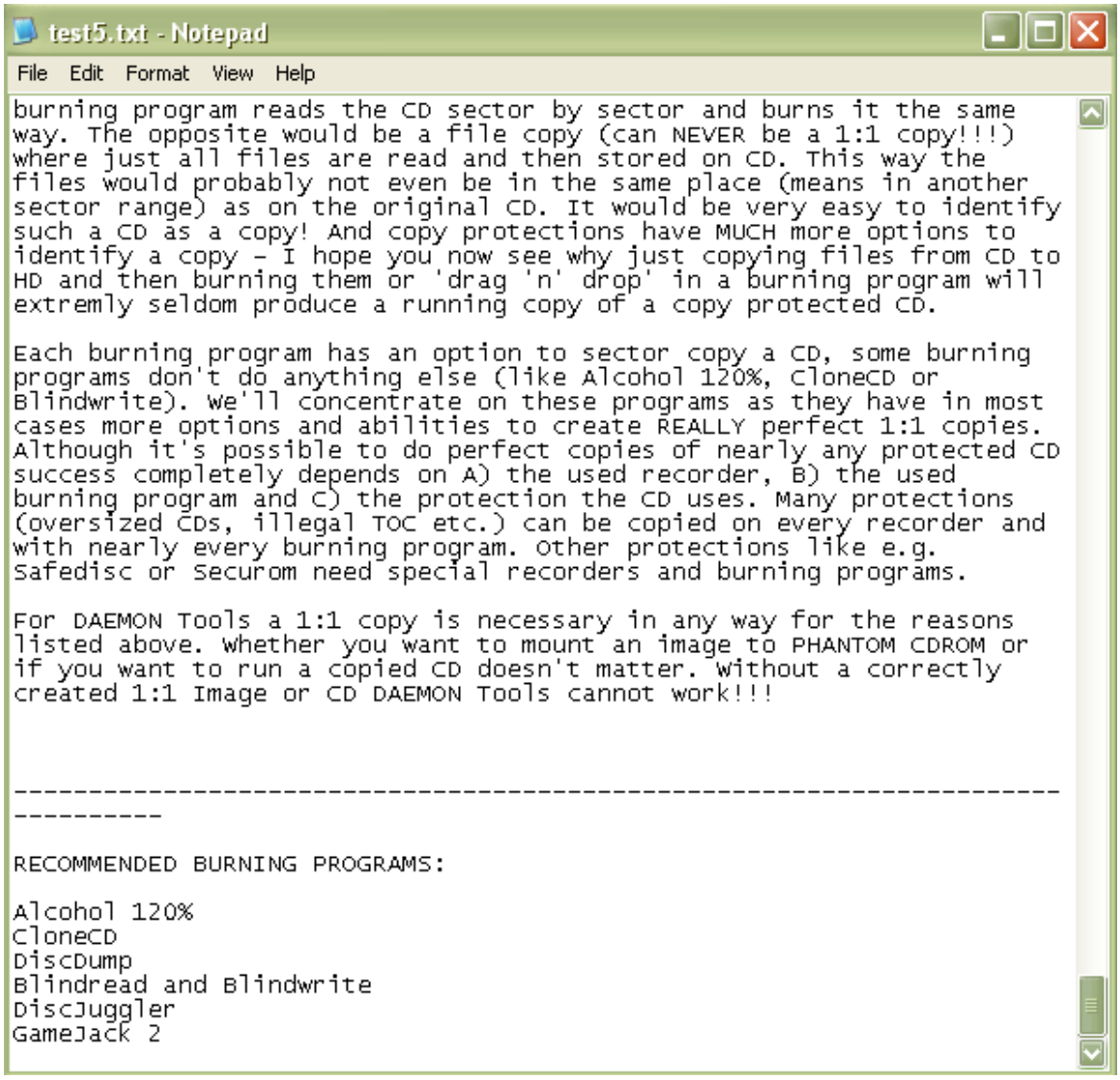

test3.txt - Notepad
File Edit Format View Help
saya mau makan..
saya mau makan kentang..
saya mau makan kentang goreng..
saya mau makan kentang goreng saja..
saya mau makan..
saya mau makan kentang..
saya mau makan kentang goreng..
saya mau makan kentang goreng saja..

4. test4.txt

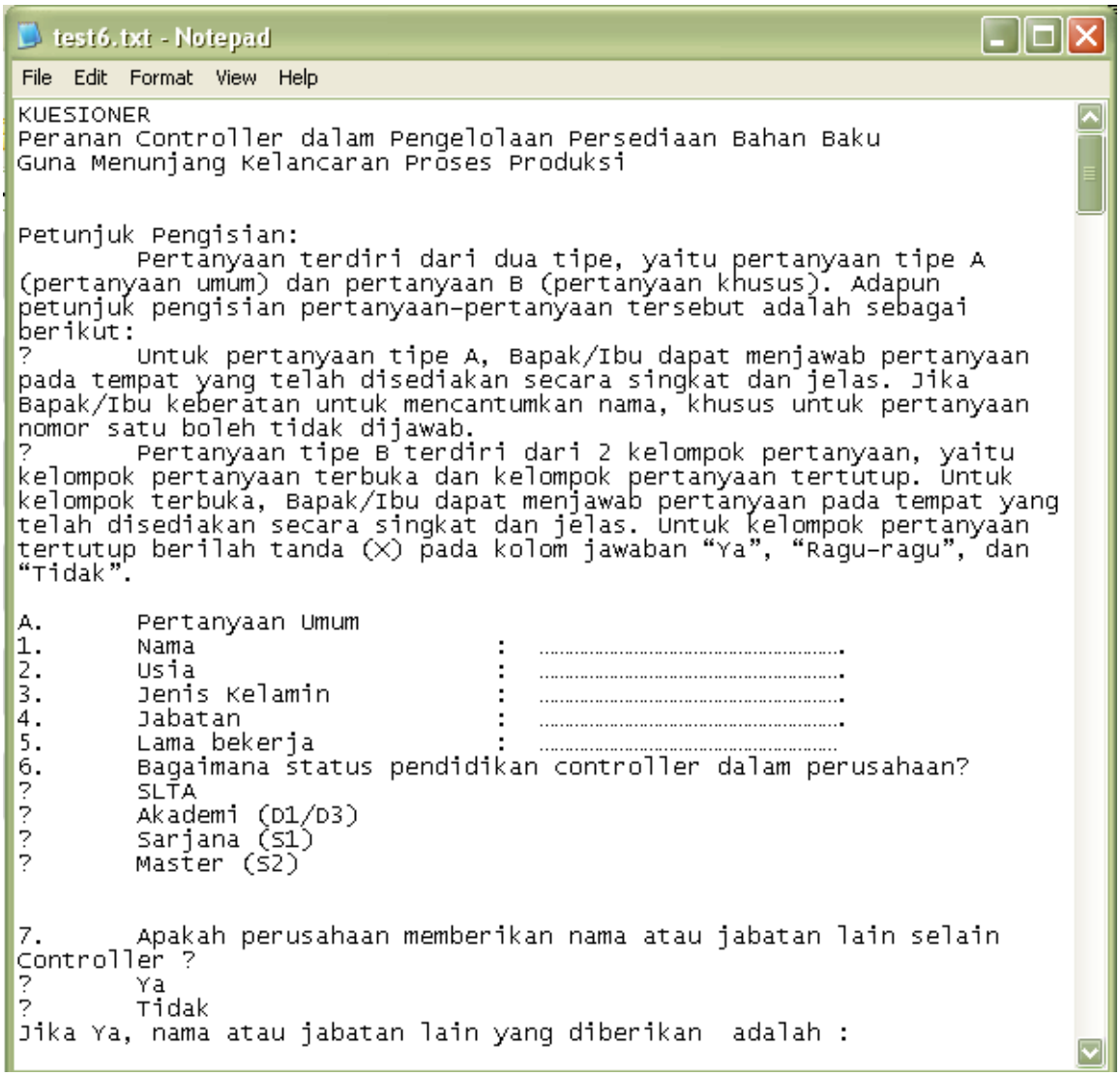

test4.txt - Notepad
File Edit Format View Help
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaa

5. **test5.txt**


test5.txt - Notepad
File Edit Format View Help
FREQUENTLY ASKED QUESTIONS:
What is DAEMON Tools?
Why does dumping an image of protected games take hours on my PC?
Why does the game xyz not run although Daemon is installed correctly?
OR
I have mounted the image of game xyz correctly, but when i start the game it asks me for the original CD. What i'm doing wrong?
Why does the Safedisc 2 emulation not work?
I downloaded an ISO of the game "XXX" from the web and it does not work with DT! What's up?
DAEMON Tools cannot open my image! Why?
How can i make an ISO Image?
When i try to make an image the program which creates the image stops with an error - Is there a way to bypass that problem?
I have a perfect 1:1 raw image mounted with DAEMON and also installed from this image but the program still requests "Insert CD"!
I made an image with a software and want to burn it with a second one, but the burning program doesn't recognize the image - How can I cope with that?
My CD writer is not able to burn in RAW mode and/or doesn't write (full) subchannel data. Can I create a working backup anyway?
I have a perfect image which works great when mounted with DAEMON Tools - but if I burnt it to CD the program refuses to work from the copy. I tried all the emulation options but nothing helped - how do I create a backup copy?
Is CDRDAO compatible with DAEMON?


test5.txt - Notepad
File Edit Format View Help
burning program reads the CD sector by sector and burns it the same way. The opposite would be a file copy (can NEVER be a 1:1 copy!!!) where just all files are read and then stored on CD. This way the files would probably not even be in the same place (means in another sector range) as on the original CD. It would be very easy to identify such a CD as a copy! And copy protections have MUCH more options to identify a copy - I hope you now see why just copying files from CD to HD and then burning them or 'drag 'n' drop' in a burning program will extremly seldom produce a running copy of a copy protected CD.
Each burning program has an option to sector copy a CD, some burning programs don't do anything else (like Alcohol 120%, CloneCD or Blindwrite). We'll concentrate on these programs as they have in most cases more options and abilities to create REALLY perfect 1:1 copies. Although it's possible to do perfect copies of nearly any protected CD success completely depends on A) the used recorder, B) the used burning program and C) the protection the CD uses. Many protections (oversized CDs, illegal TOC etc.) can be copied on every recorder and with nearly every burning program. Other protections like e.g. Safedisc or Securom need special recorders and burning programs.
For DAEMON Tools a 1:1 copy is necessary in any way for the reasons listed above. Whether you want to mount an image to PHANTOM CDROM or if you want to run a copied CD doesn't matter. Without a correctly created 1:1 Image or CD DAEMON Tools cannot work!!!
RECOMMENDED BURNING PROGRAMS:
Alcohol 120%
CloneCD
DiscDump
Blindread and Blindwrite
DiscJuggler
GameJack 2

6. test6.txt



test6.txt - Notepad

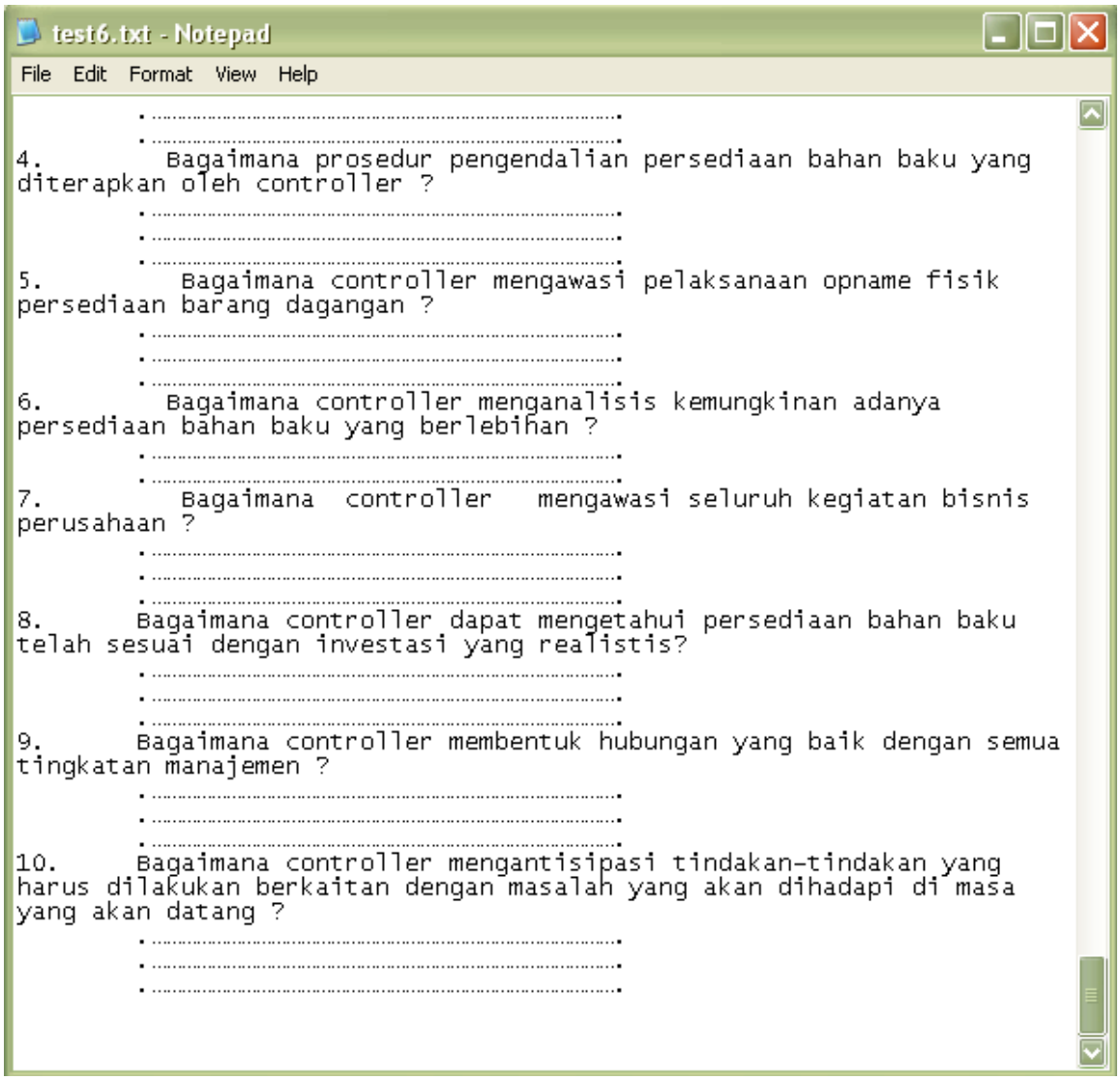
File Edit Format View Help

KUESIONER
Peranan Controller dalam Pengelolaan Persediaan Bahan Baku
Guna Menunjang Kelancaran Proses Produksi

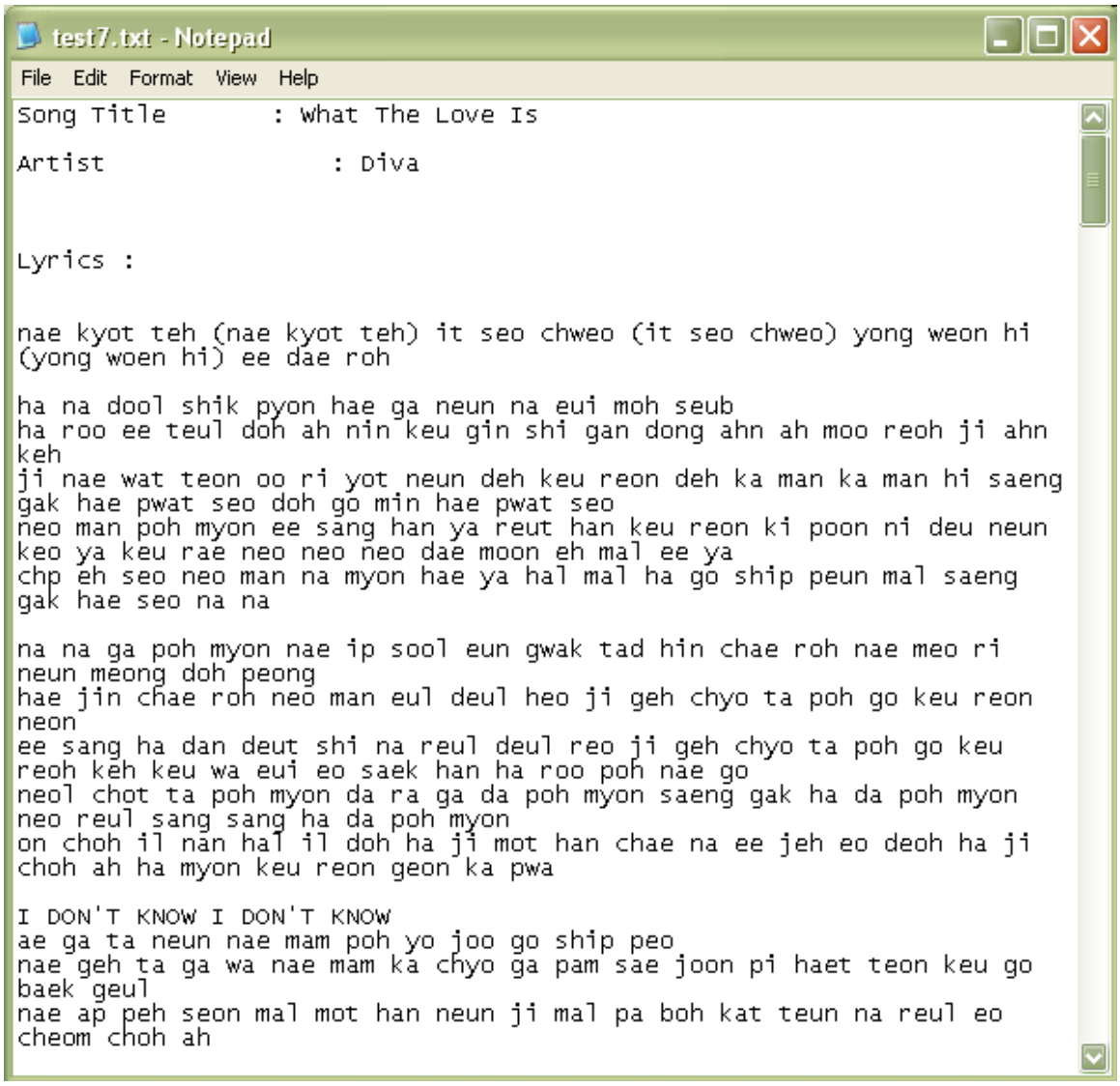
Petunjuk Pengisian:
Pertanyaan terdiri dari dua tipe, yaitu pertanyaan tipe A (pertanyaan umum) dan pertanyaan B (pertanyaan khusus). Adapun petunjuk pengisian pertanyaan-pertanyaan tersebut adalah sebagai berikut:
? Untuk pertanyaan tipe A, Bapak/Ibu dapat menjawab pertanyaan pada tempat yang telah disediakan secara singkat dan jelas. Jika Bapak/Ibu keberatan untuk mencantumkan nama, khusus untuk pertanyaan nomor satu boleh tidak dijawab.
? Pertanyaan tipe B terdiri dari 2 kelompok pertanyaan, yaitu kelompok pertanyaan terbuka dan kelompok pertanyaan tertutup. Untuk kelompok terbuka, Bapak/Ibu dapat menjawab pertanyaan pada tempat yang telah disediakan secara singkat dan jelas. Untuk kelompok pertanyaan tertutup berilah tanda (X) pada kolom jawaban "Ya", "Ragu-ragu", dan "Tidak".

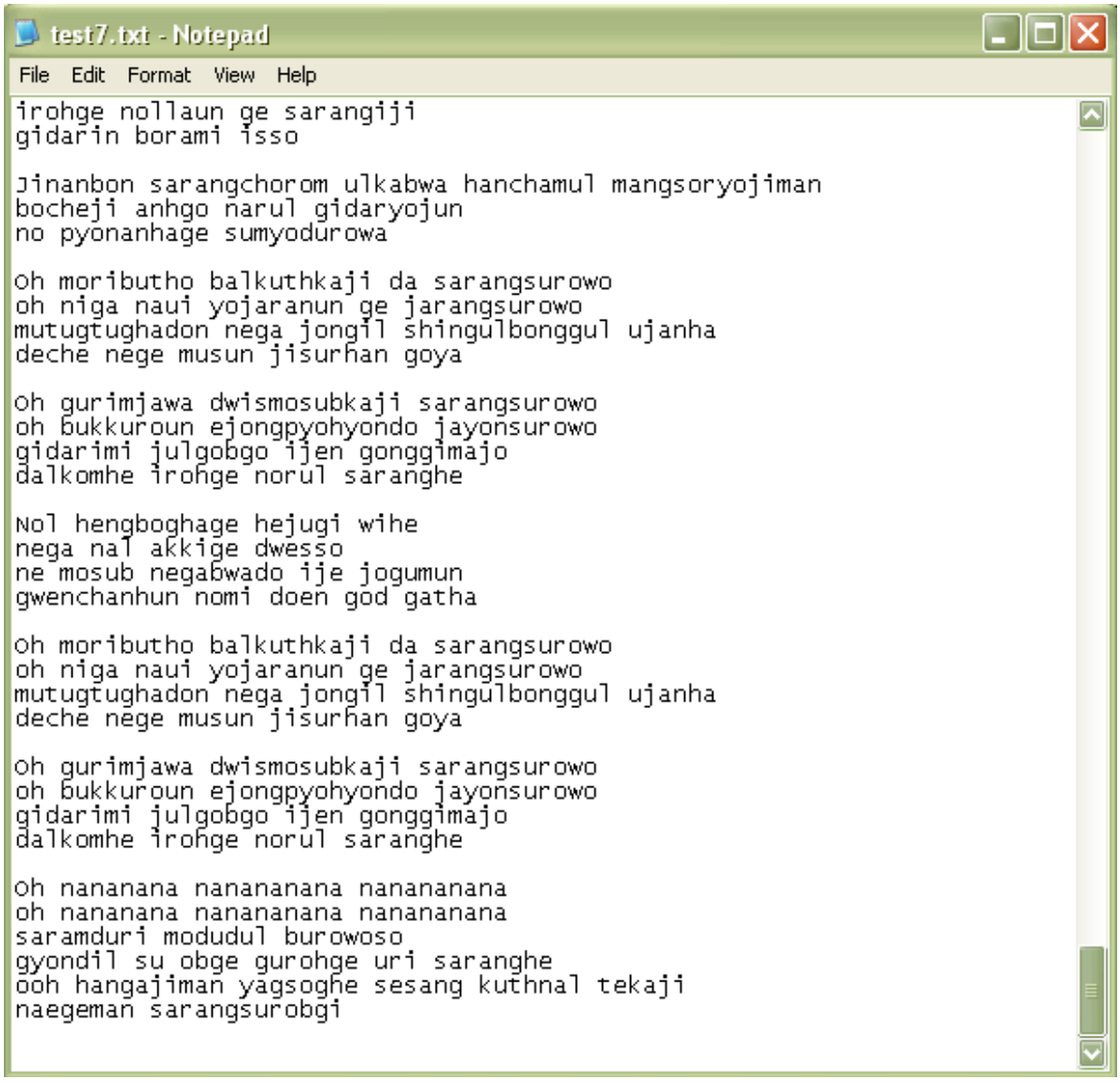
A. Pertanyaan Umum
1. Nama : .....
2. Usia : .....
3. Jenis Kelamin : .....
4. Jabatan : .....
5. Lama bekerja : .....
6. Bagaimana status pendidikan controller dalam perusahaan?
? SLTA
? Akademi (D1/D3)
? Sarjana (S1)
? Master (S2)

7. Apakah perusahaan memberikan nama atau jabatan lain selain Controller ?
? Ya
? Tidak
Jika Ya, nama atau jabatan lain yang diberikan adalah :


test6.txt - Notepad
File Edit Format View Help
4. Bagaimana prosedur pengendalian persediaan bahan baku yang diterapkan oleh controller ?
5. Bagaimana controller mengawasi pelaksanaan opname fisik persediaan barang dagangan ?
6. Bagaimana controller menganalisis kemungkinan adanya persediaan bahan baku yang berlebihan ?
7. Bagaimana controller mengawasi seluruh kegiatan bisnis perusahaan ?
8. Bagaimana controller dapat mengetahui persediaan bahan baku telah sesuai dengan investasi yang realistis?
9. Bagaimana controller membentuk hubungan yang baik dengan semua tingkatan manajemen ?
10. Bagaimana controller mengantisipasi tindakan-tindakan yang harus dilakukan berkaitan dengan masalah yang akan dihadapi di masa yang akan datang ?

7. test7.txt


test7.txt - Notepad
File Edit Format View Help
Song Title : What The Love Is
Artist : Diva
Lyrics :
nae kyot teh (nae kyot teh) it seo chweo (it seo chweo) yong weon hi (yong woen hi) ee dae roh
ha na dool shik pyon hae ga neun na eui moh seub
ha roo ee teul doh ah nin keu gin shi gan dong ahn ah moo reoh ji ahn keh
ji nae wat teon oo ri yot neun deh keu reon deh ka man ka man hi saeng gak hae pwat seo doh go min hae pwat seo
neo man poh myon ee sang han ya reut han keu reon ki poon ni deu neun keo ya keu rae neo neo neo dae moon eh mal ee ya
chp eh seo neo man na myon hae ya hal mal ha go ship peun mal saeng gak hae seo na na
na na ga poh myon nae ip sool eun gwak tad hin chae roh nae meo ri neun meong doh peong
hae jin chae roh neo man eul deul heo ji geh chyo ta poh go keu reon neon
ee sang ha dan deut shi na reul deul reo ji geh chyo ta poh go keu reoh keh keu wa eui eo saek han ha roo poh nae go
neol chot ta poh myon da ra ga da poh myon saeng gak ha da poh myon
neo reul sang sang ha da poh myon
on choh il nan hal il doh ha ji mot han chae na ee jeh eo deoh ha ji choh ah ha myon keu reon geon ka pwa
I DON'T KNOW I DON'T KNOW
ae ga ta neun nae mam poh yo joo go ship peo
nae geh ta ga wa nae mam ka chyo ga pam sae joon pi haet teon keu go baek geul
nae ap peh seon mal mot han neun ji mal pa boh kat teun na reul eo cheom choh ah


test7.txt - Notepad
File Edit Format View Help
irohge nollaun ge sarangiji
gidarin borami isso

Jinanbon sarangchorom ulkabwa hanchamul mangsoryojiman
bocheji anhgo narul gidaryojun
no pyonanhage sumyodurowa

Oh moributho balkuthkaji da sarangsurowo
oh niga naui yojaranun ge jarangsurowo
mutugtughadon nega jongil shingulbonggul ujanha
deche nege musun jisurhan goya

Oh gurimjawa dwismosubkaji sarangsurowo
oh bukkuroun ejongpyohyondo jayonsurowo
gidarimi julgobgo ijen gonggimajo
dalkomhe irohge norul saranghe

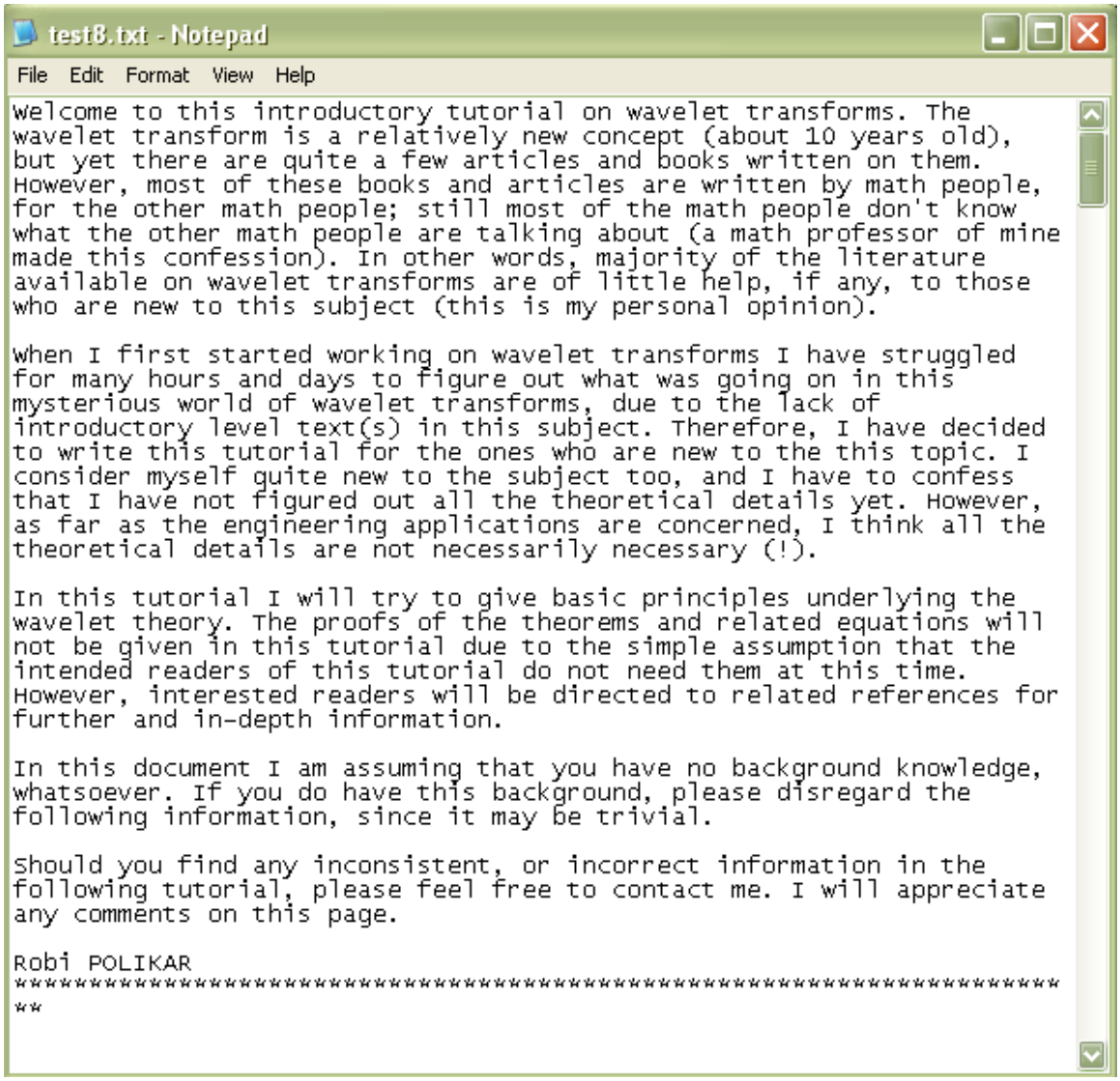
Nol hengboghage hejugi wihe
nega nal akkige dwesso
ne mosub negabwado ije jogumun
gwenchanhun nomi doen god gatha

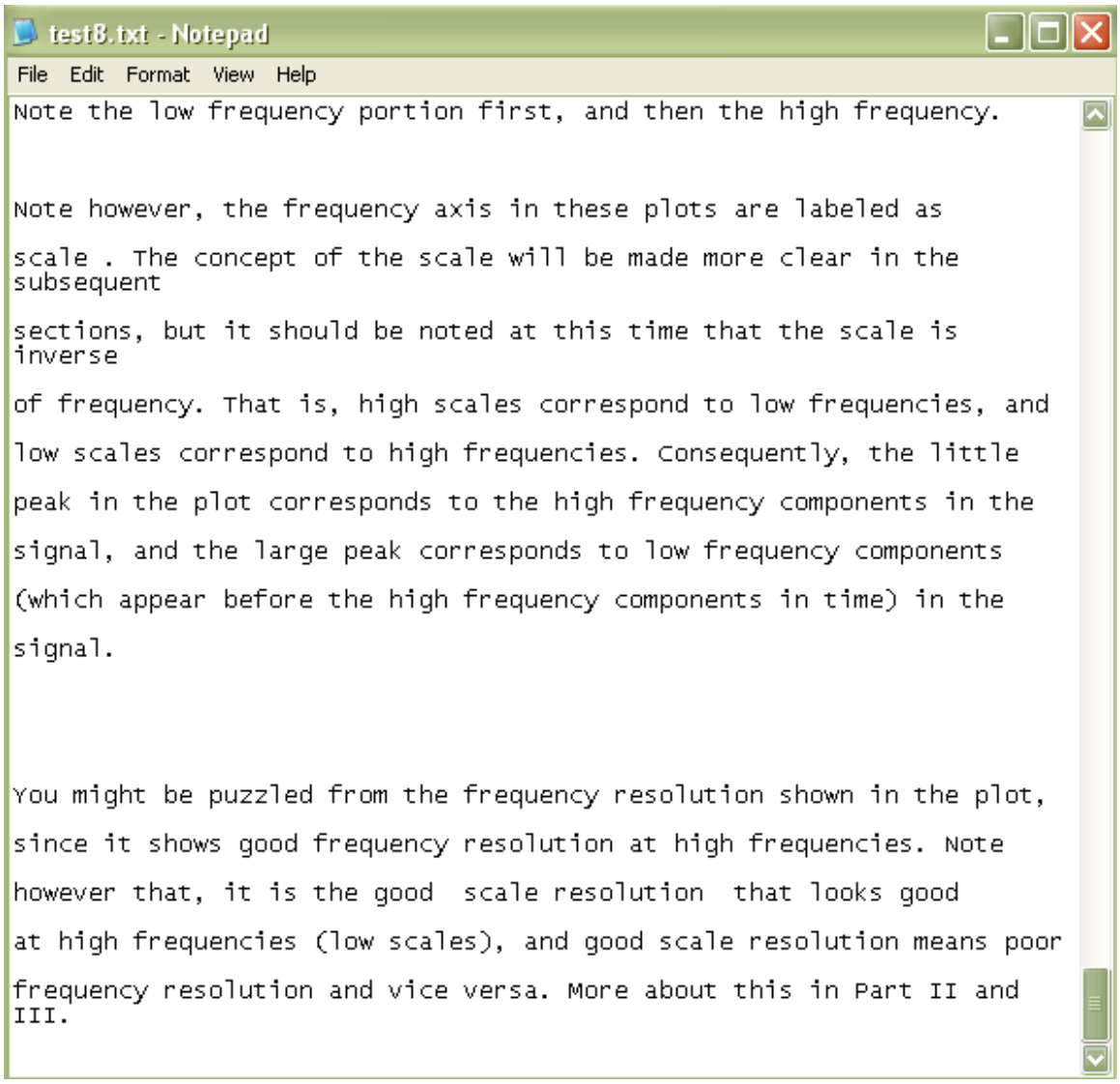
Oh moributho balkuthkaji da sarangsurowo
oh niga naui yojaranun ge jarangsurowo
mutugtughadon nega jongil shingulbonggul ujanha
deche nege musun jisurhan goya

Oh gurimjawa dwismosubkaji sarangsurowo
oh bukkuroun ejongpyohyondo jayonsurowo
gidarimi julgobgo ijen gonggimajo
dalkomhe irohge norul saranghe

Oh nananana nanananana nanananana
oh nananana nanananana nanananana
saramduri modudul burowoso
gyondil su obge gurohge uri saranghe
ooh hangajiman yagsoghe sesang kuthnal tekaji
naegeman sarangsurobgi

8. test8.txt


test8.txt - Notepad
File Edit Format View Help
welcome to this introductory tutorial on wavelet transforms. The wavelet transform is a relatively new concept (about 10 years old), but yet there are quite a few articles and books written on them. However, most of these books and articles are written by math people, for the other math people; still most of the math people don't know what the other math people are talking about (a math professor of mine made this confession). In other words, majority of the literature available on wavelet transforms are of little help, if any, to those who are new to this subject (this is my personal opinion).
When I first started working on wavelet transforms I have struggled for many hours and days to figure out what was going on in this mysterious world of wavelet transforms, due to the lack of introductory level text(s) in this subject. Therefore, I have decided to write this tutorial for the ones who are new to the this topic. I consider myself quite new to the subject too, and I have to confess that I have not figured out all the theoretical details yet. However, as far as the engineering applications are concerned, I think all the theoretical details are not necessarily necessary (!).
In this tutorial I will try to give basic principles underlying the wavelet theory. The proofs of the theorems and related equations will not be given in this tutorial due to the simple assumption that the intended readers of this tutorial do not need them at this time. However, interested readers will be directed to related references for further and in-depth information.
In this document I am assuming that you have no background knowledge, whatsoever. If you do have this background, please disregard the following information, since it may be trivial.
Should you find any inconsistent, or incorrect information in the following tutorial, please feel free to contact me. I will appreciate any comments on this page.
Robi POLIKAR
****
**

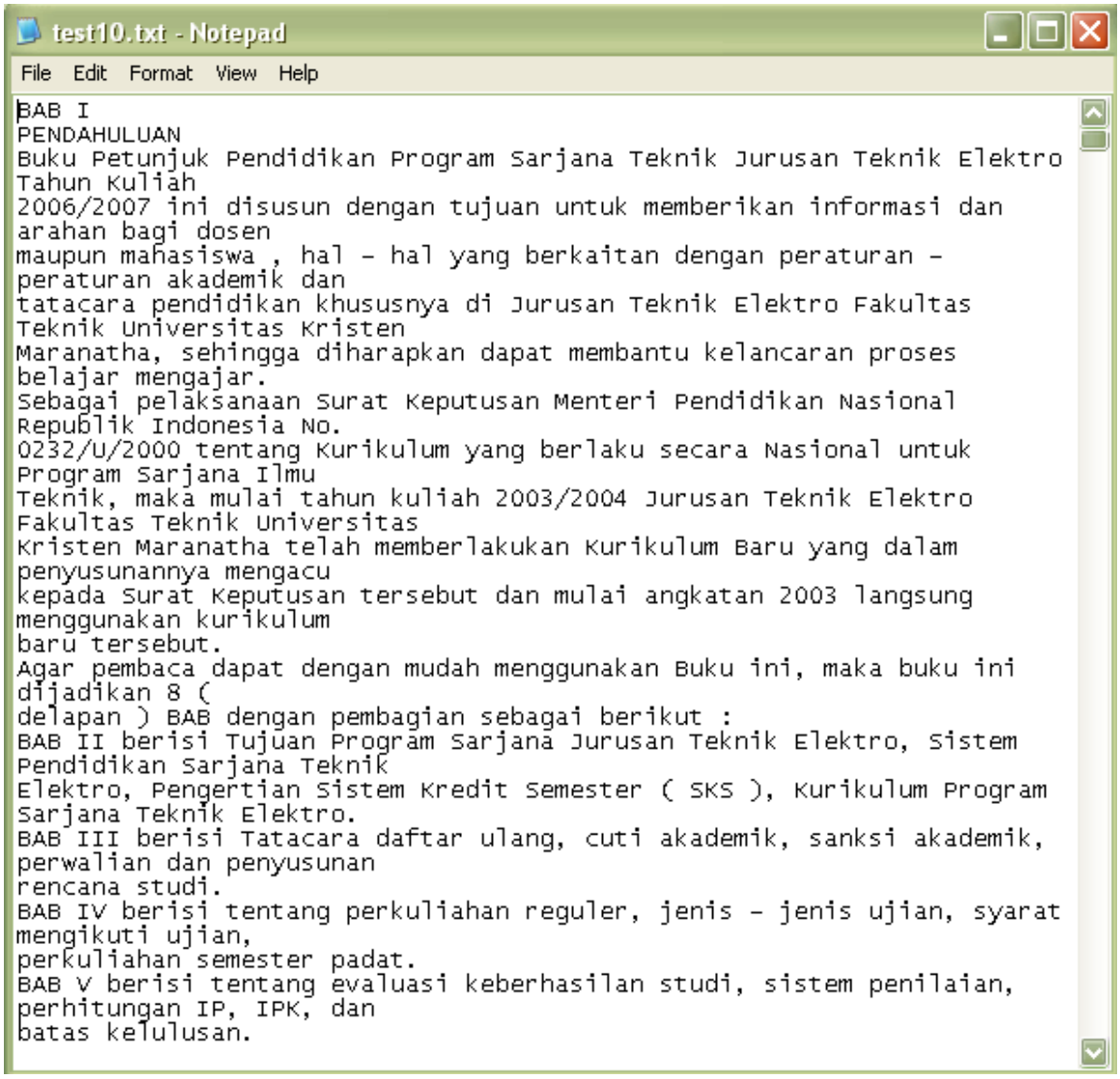

test8.txt - Notepad
File Edit Format View Help
Note the low frequency portion first, and then the high frequency.
Note however, the frequency axis in these plots are labeled as
scale . The concept of the scale will be made more clear in the subsequent
sections, but it should be noted at this time that the scale is inverse
of frequency. That is, high scales correspond to low frequencies, and
low scales correspond to high frequencies. Consequently, the little
peak in the plot corresponds to the high frequency components in the
signal, and the large peak corresponds to low frequency components
(which appear before the high frequency components in time) in the
signal.
You might be puzzled from the frequency resolution shown in the plot,
since it shows good frequency resolution at high frequencies. Note
however that, it is the good scale resolution that looks good
at high frequencies (low scales), and good scale resolution means poor
frequency resolution and vice versa. More about this in Part II and
III.

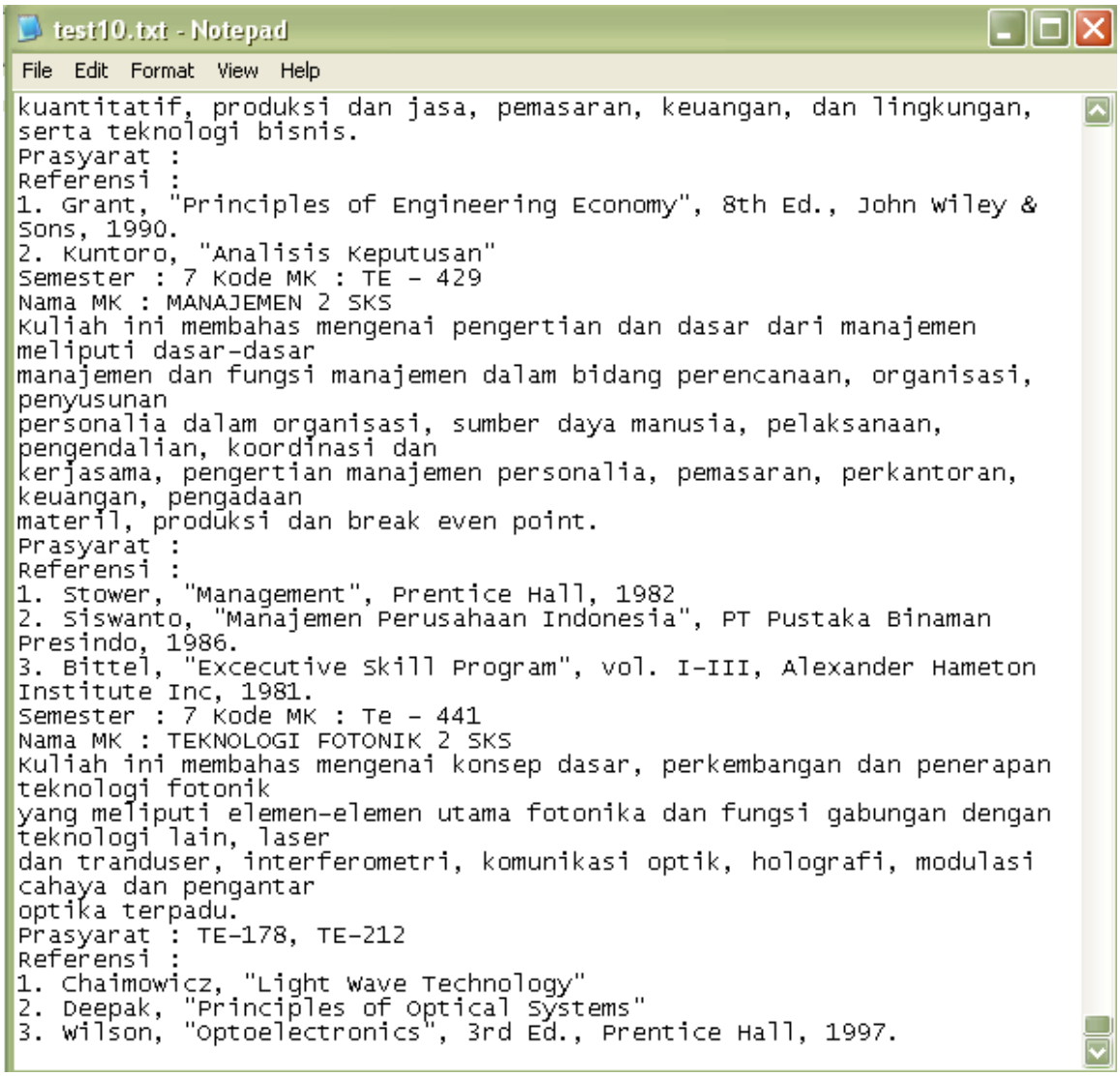
9. test9.txt


test9.txt - Notepad
File Edit Format View Help
Abstract :
WiMAX is defined as Worldwide Interoperability for Microwave Access by the WiMAX Forum [1], formed in April 2001 to promote conformance and interoperability of the standard IEEE 802.16. The Forum describes WiMAX as "a standards-based technology enabling the delivery of last mile wireless broadband access as an alternative to cable and DSL."The WiMAX Forum is "the exclusive organization dedicated to certifying the interoperability of BWA products, the WiMAX Forum defines and conducts conformance and interoperability testing to ensure that different vendor systems work seamlessly with one another."Those that pass conformance and interoperability testing achieve the "WiMAX Forum Certified" designation and display this mark on their products and marketing materials. Vendors claiming their equipment is "WiMAX-ready," "WiMAX-compliant," "pre-WiMAX," are not WiMAX Forum Certified, according to the Forum. [2]The IEEE 802.16 media access controller (MAC) is significantly different from that of IEEE 802.11 Wi-Fi MAC. In Wi-Fi, the MAC uses contention access—all subscriber stations wishing to pass data through an access point are competing for the AP's attention on a random basis. This can cause distant nodes from the AP to be repeatedly interrupted by less sensitive, closer nodes, greatly reducing their throughput. And this makes services, such as VoIP or IPTV which depend on a determined level of quality of service (QoS) difficult to maintain for large numbers of users.In contrast, the 802.16 MAC is a scheduling MAC where the subscriber station only has to compete once (for initial entry into the network). After that it is allocated a time slot by the base station. The time slot can enlarge and constrict, but it remains assigned to the subscriber station meaning that other subscribers are not supposed to use it but take their turn. This scheduling algorithm is stable under overload and over-subscription (unlike 802.11). It is also much more bandwidth efficient. The scheduling algorithm also allows the base station to control Quality of Service by balancing the assignments among the needs of the subscriber stations.A recent addition to the WiMAX standard is underway which will add full mesh networking capability by enabling WiMAX nodes to simultaneously operate in "subscriber station" and "base station" mode. This will blur that initial distinction and allow for widespread adoption of WiMAX based mesh networks and promises widespread WiMAX adoption. WiMAX/802.16's use of OFDMA and scheduled MAC allows wireless mesh networks to be


test9.txt - Notepad
File Edit Format View Help
Intel malahan berani menjanjikan terintegrasinya teknologi modem dan antena pada chip Centrino di akhir tahun 2006 nanti. Modem WiMAX yang akan beredar di pasaran tahun depan dijual dengan harga pada kisaran 500 dolar AS. Namun, tiga tahun ke depan harga ini akan turun hingga 10% saja. Prediksi para analis juga mengatakan, penjualan WiMAX pada tiga tahun ke depan ini akan meningkat enam kali lipat. Ini mudah-mudahan bisa menjadi katalis serupa untuk Indonesia.Isu lainnya yang mungkin menarik adalah mengenai pemakaian frekuensi. Apakah akan muncul kembali kericuhan seperti yang terjadi selama ini dengan izin pemakaian frekuensi tersebut? Ini bisa menjadi entry-barrier yang cukup potensial bagi meluasnya pemakaian WiMAX di Indonesia.Tidak kalah pula pentingnya adalah masalah investasi jaringan base-station. Penyelenggara telefon seluler mungkin dinilai memiliki start awal dengan kesiapan base-station mereka.Tapi, boleh jadi sebenarnya ada kekhawatiran mengenai berpeluangnya WiMAX untuk menganibalisasi layanan-layanan yang diberikan telefon seluler tersebut. Yang menjadi pertimbangan barangkali, seberapa jauhkah operator seluler mau melepaskan sebagian revenue mereka kepada bisnis WiMAX ini? Saat ini, WiMAX masih berada dalam tahap uji coba di banyak negara, termasuk Amerika, Inggris, Cina, dan India.
3. Kesimpulan
WiMAX tidak hanya cocok sebagai solusi komunikasi data namun juga cocok sebagai alternative untuk komunikasi.Daya jangkau dan kapasitas mengirimkan data yang besar adalah kekuatan WiMAX dibandingkan teknologi sebelumnya.WiMAX juga mampu mengirimkan data hingga 75 megabit per detik untuk setiap base station(BTS) dengan jari jari sel berukuran 2 hingga 10 kilometer dan juga standar IEEE 802.16 memberikan kemudahan dalam akses internet untuk areametropolitan dengan menerapkan beberapa base station (BS) yang dapat mengcoveragejutaan subscriber station (SS).
Akhir-akhir ini teknologi wireless telah berkembang sangat pesat yangmenyediakan hubungan telekomunikasi tanpa tergantung dari jaringan kabel menggunakan telepon atau TV kabel. Dengan adanya teknologi wirelessMANTM maka akan memberikan kemurahan dan kecepatan transfer data dengan akses
internet jalur lebar (broadband).

10. test10.txt




test10.txt - Notepad
File Edit Format View Help
kuantitatif, produksi dan jasa, pemasaran, keuangan, dan lingkungan, serta teknologi bisnis.
Prasyarat :
Referensi :
1. Grant, "Principles of Engineering Economy", 8th Ed., John Wiley & Sons, 1990.
2. Kuntoro, "Analisis Keputusan"
Semester : 7 Kode MK : TE - 429
Nama MK : MANAJEMEN 2 SKS
Kuliah ini membahas mengenai pengertian dan dasar dari manajemen meliputi dasar-dasar
manajemen dan fungsi manajemen dalam bidang perencanaan, organisasi, penyusunan
personalia dalam organisasi, sumber daya manusia, pelaksanaan, pengendalian, koordinasi dan
kerjasama, pengertian manajemen personalia, pemasaran, perkantoran, keuangan, pengadaan
materil, produksi dan break even point.
Prasyarat :
Referensi :
1. Stower, "Management", Prentice Hall, 1982
2. Siswanto, "Manajemen Perusahaan Indonesia", PT Pustaka Binaman Presindo, 1986.
3. Bittel, "Excecutive Skill Program", vol. I-III, Alexander Hameton Institute Inc, 1981.
Semester : 7 Kode MK : Te - 441
Nama MK : TEKNOLOGI FOTONIK 2 SKS
Kuliah ini membahas mengenai konsep dasar, perkembangan dan penerapan teknologi fotonik
yang meliputi elemen-elemen utama fotonika dan fungsi gabungan dengan teknologi lain, laser
dan tranduser, interferometri, komunikasi optik, holografi, modulasi cahaya dan pengantar
optika terpadu.
Prasyarat : TE-178, TE-212
Referensi :
1. Chaimowicz, "Light Wave Technology"
2. Deepak, "Principles of Optical Systems"
3. Wilson, "Optoelectronics", 3rd Ed., Prentice Hall, 1997.