LAMPIRAN

LAMPIRAN A
LISTING PROGRAM

//Program Utama

```
unit uMain;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, Menus, ExtCtrls, StdCtrls,
  Lucifer,VisSem, Spin, ZipForge;

type
  TDisplayProc = procedure(const s: string;Dest:TMemo) of object;
  TfrmMain = class(TForm)
    mnu1: TMainMenu;
    Menu11: TMenuItem;
    Menu21: TMenuItem;
    pcUtama: TPageControl;
    tsEnkrip: TTabSheet;
    tsDekrip: TTabSheet;
    edCarrEnc: TLabeledEdit;
    btnCarrBrwEnc: TButton;
    lblSizeEnc: TLabel;
    edPassEnc: TLabeledEdit;
    pcEnc: TPageControl;
    tsEncText: TTabSheet;
    tsEncFile: TTabSheet;
    edTextEnc: TLabeledEdit;
    btnEncText: TButton;
```

edFileSisip: TLabeledEdit;
btnBrwSisip: TButton;
lblSizeSisip: TLabel;
memFileToSisip: TMemo;
Label3: TLabel;
btnProcEnc: TButton;
chkCompr: TCheckBox;
btnSaveEnc: TButton;
Label5: TLabel;
memFileTersisip: TMemo;
btnCarrBrwDec: TButton;
edCarrDec: TLabeledEdit;
edPassDec: TLabeledEdit;
memCarrDec: TMemo;
memResult: TMemo;
btnSaveDec: TButton;
chkDeCompr: TCheckBox;
memCarrEnc: TMemo;
Label8: TLabel;
Label9: TLabel;
Label10: TLabel;
memFileEncText: TMemo;
Label12: TLabel;
btnProcDec: TButton;
odMedium: TOpenDialog;
odSisip: TOpenDialog;
sdResult: TSaveDialog;
btnSaveText: TButton;
sdText: TSaveDialog;
spMaxEnc: TSpinEdit;
Label1: TLabel;
lblPjgText: TLabel;

```
    zf1: TZipForge;
    memLastResult: TMemo;
    Label7: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure btnCloseClick(Sender: TObject);
    procedure ClearTabs(Sender: TObject);
    procedure btnCarrBrwEncClick(Sender: TObject);
    procedure btnEncTextClick(Sender: TObject);
    procedure edTextEncChange(Sender: TObject);
    procedure btnSaveTextClick(Sender: TObject);
    procedure btnBrwSisipClick(Sender: TObject);
    procedure btnProcEncClick(Sender: TObject);
    procedure btnCarrBrwDecClick(Sender: TObject);
    procedure btnSaveEncClick(Sender: TObject);
    procedure btnProcDecClick(Sender: TObject);
    procedure btnSaveDecClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure pcUtamaChanging(Sender: TObject; var AllowChange: Boolean);
    procedure pcEncChanging(Sender: TObject; var AllowChange: Boolean);
    procedure SwitchTab(Sender :TObject);
  private
    { Private declarations }
    Encoded, Zipped,Unknown : String;
    saved : boolean;
    procedure Display(const S: String;Dest:TMemo);
    procedure ShowBinary(var Data; Count: Cardinal; DispProc:
TDisplayProc;Dest:TMemo;MaxLen:Cardinal);
    function IsTextFile(const sFile: TFileName): boolean;
  public
    { Public declarations }
  end;
```

```
var
  frmMain: TfrmMain;
  SizeMedium, SizeFileSisip, SizeTeks : Integer;

implementation

{$R *.dfm}

procedure TfrmMain.FormCreate(Sender: TObject);
begin
  pcUtama.ActivePageIndex := 0;
  //rgMode.ItemIndex := 0;
  pcEnc.ActivePageIndex := 0;
  ClearTabs(Self);
  Encoded := IntToStr(GetTickCount);// + '.enc';
  Sleep(100);
  Zipped := IntToStr(GetTickCount);// + '.zpd';
  Sleep(100);
  Unknown := IntToStr(GetTickCount);
end;

procedure TfrmMain.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  DeleteFile(Zipped);
  DeleteFile(Encoded);
  DeleteFile(Unknown);
end;

procedure TfrmMain.btnCloseClick(Sender: TObject);
begin
  Close;
end;
```

```
procedure TfrmMain.ShowBinary(var Data; Count: Cardinal;
  DispProc: TDisplayProc;Dest:TMemo;MaxLen:Cardinal);
var
  line: string[80];
  i: Cardinal;
  p: PChar;
const
  ascStart = 1;
  HexChars: PChar = '0123456789ABCDEF';
begin
  p    := @Data;
  line := '';
  for i := 0 to Count - 1 do
  begin
    if (i mod MaxLen) = 0 then
    begin
      if Length(line) > 0 then
        DispProc(line,Dest);
      FillChar(line, SizeOf(line), ' ');
    end;
    if p[i] >= ' ' then
      line[i mod MaxLen + ascStart] := p[i]
    else
      line[i mod MaxLen + ascStart] := '.';
  end;
  DispProc(line,Dest);
end;

procedure TfrmMain.Display(const S: String;Dest:TMemo);
begin
  Dest.Lines.Add(S);
```

```
end;

function TfrmMain.IsTextFile(const sFile: TFileName): boolean;
var
 oIn: TFileStream;
 iRead: Int64;
 iData: Byte;
 Selesai : boolean;
begin
 selesai := False;
 iRead := 1;
 oIn := TFileStream.Create(sFile, fmOpenRead or fmShareDenyNone);
 try
  While (not Selesai) and (iRead < oIn.Size) do
   begin
    oIn.Read(iData, 1);
    if (iData <> 13) and (iData <> 9) and (iData <> 10) then
     Selesai := ((iData < 32) or (iData > 127));
    inc(iRead);
   end;
  Result := Not(Selesai);
 finally
   FreeAndNil(oIn);
 end;
end;

procedure TfrmMain.SwitchTab(Sender : TObject);
begin
//  tsEncText.TabVisible := Boolean(Teks);
//  tsEncFile.TabVisible := Boolean(Files);
  edTextEnc.Clear;
  memFileEncText.Lines.Clear;
```

```
  btnSaveText.Enabled := False;
  edFileSisip.Clear;
  lblSizeSisip.Caption := 'Ukuran file yang akan dsisip : 0 Bytes';
  memFileToSisip.Lines.Clear;
  memFileTersisip.Lines.Clear;
  chkCompr.Checked := False;
  btnSaveDec.Enabled := False;
  lblPjgText.Caption := 'Panjang Teks : 0 Karakter';
end;

procedure TfrmMain.ClearTabs(Sender: TObject);
begin
  SizeMedium := 0;
  SizeFileSisip := 0;
  SizeTeks := 0;
  edCarrEnc.Clear;
  edPassEnc.Clear;
  lblSizeEnc.Caption := 'Ukuran Carrier : 0 bits, 0 Bytes';
  pcEnc.ActivePageIndex := 0;
  edTextEnc.Clear;
  memFileEncText.Lines.Clear;
  edFileSisip.Clear;
  lblSizeSisip.Caption := 'Ukuran file yang akan dsisip : 0 Bytes';
  lblPjgText.Caption := 'Panjang Teks : 0 Bytes';
  chkCompr.Checked := False;
  memFileToSisip.Lines.Clear;
  memFileTersisip.Lines.Clear;
  memCarrEnc.Lines.Clear;
  btnSaveEnc.Enabled := False;
  btnSaveText.Enabled := False;
  spMaxEnc.Value := 50;
```

```
  edCarrDec.Clear;
  edPassDec.Clear;
  memCarrDec.Lines.Clear;
  memResult.Lines.Clear;
  memLastResult.Lines.Clear;
  chkDeCompr.Checked := False;
  btnSaveDec.Enabled := False;

  DeleteFile(Unknown);
  DeleteFile(Encoded);
  DeleteFile(Zipped);

end;

procedure TfrmMain.btnCarrBrwEncClick(Sender: TObject);
begin
  edCarrEnc.Clear;
  edPassEnc.Clear;
  edTextEnc.Clear;
  lblSizeEnc.Caption := 'Ukuran Carrier : 0 bits, 0 Bytes';
  memCarrEnc.Lines.Clear;
  memFileEncText.Lines.Clear;
  btnSaveText.Enabled := False;
  edFileSisip.Clear;
  lblSizeSisip.Caption := 'Ukuran file yang akan dsisip : 0 Bytes';
  memFileToSisip.Lines.Clear;
  memFileTersisip.Lines.Clear;
  chkCompr.Checked := False;
  odMedium.FileName := '';
  Limit := spMaxEnc.Value;
  if odMedium.Execute then
   begin
```

```
    edCarrEnc.Text := odMedium.FileName;
    SizeMedium := HitungKapasitas(edCarrEnc.Text);
    lblSizeEnc.Caption := Format('Ukuran Carrier : %d bits, %d Bytes',
     [SizeMedium,SizeMedium div 16]);
    SizeMedium := SizeMedium div 16;
    memCarrEnc.Lines.LoadFromFile(edCarrEnc.Text);
    memCarrEnc.Lines.SaveToFile(Encoded);
    odMedium.FileName := '';
  end;
end;

procedure TfrmMain.edTextEncChange(Sender: TObject);
var TotLen : Integer;
begin
  if edTextEnc.Text <> '' then
   TotLen := 16 * (((Length(edTextEnc.Text)-1) div 16)+1)
  else
   TotLen := 0;
  lblPjgText.Caption := Format('Panjang Teks : %d Bytes',
   [TotLen])
end;

procedure TfrmMain.btnEncTextClick(Sender: TObject);
var StrResult : String;
begin
  SizeTeks := 0;
  MemFileEncText.Lines.Clear;
  if (edPassEnc.Text <> '') and (Length(EdPassEnc.Text) = 16)
   and (edTextEnc.Text <> '') and (edCarrEnc.Text <> '')then
   begin
    SizeTeks := 16 * (((Length(edTextEnc.Text)-1) div 16)+1);
    if SizeTeks <= SizeMedium then
```

```
      begin
        EncDec(edPassEnc.Text,edTextEnc.Text,StrResult,0);
        Sisip(StrResult,Encoded);
        memFileEncText.Lines.LoadFromFile(Encoded);
        btnSaveText.Enabled := True;
        MessageDlg('Proses selesai',mtInformation,[mbOk],0);
      end
     else
      MessageDlg('Daya tampung medium tidak mencukupi!!',mtError,[mbOk],0);
   end
  else
   MessageDlg('Lengkapi inputan : File Carrier, Password,
Teks',mtWarning,[mbOk],0);
end;

procedure TfrmMain.btnSaveTextClick(Sender: TObject);
begin
  sdText.FileName := '';
  saved := false;
  if sdText.Execute then
   begin
     memFileEncText.Lines.SaveToFile(sdText.FileName);
     MessageDlg('File berhasil disimpan..',mtInformation,[mbOk],0);
     sdText.FileName := '';
     saved := true;
   end;
end;

procedure TfrmMain.btnBrwSisipClick(Sender: TObject);
var i,j:integer;
    FS : TMemoryStream;
begin
```

```
  odSisip.FileName := '';
  edFileSisip.Clear;
  SizeFileSisip := 0;
  lblSizeSisip.Caption := 'Ukuran file yang akan dsisip : 0 Bytes';
  memFileToSisip.Lines.Clear;
  if odSisip.Execute then
   begin
    edFileSisip.Text := odSisip.FileName;
    odSisip.FileName := '';
    i := FileOpen(edFileSisip.Text,fmOpenRead);
    SizeFileSisip := GetFileSize(i,@j);
    SizeFileSisip := SizeFileSisip + (j*-1);
    SizeFileSisip := 16 * (((SizeFileSisip - 1) div 16) + 1);
    FileClose(i);
    lblSizeSisip.Caption := Format('Ukuran file yang akan dsisip : %d Bytes',
     [SizeFileSisip]);
    FS := TMemoryStream.Create;
    FS.LoadFromFile(edFileSisip.Text);
    if IsTextFile(edFileSisip.Text) then
     memFileToSisip.Lines.LoadFromStream(FS)
    else
     ShowBinary(FS.Memory^, FS.Size, Display, memFileToSisip,30);
    FS.Free;
   end;
end;

procedure TfrmMain.btnProcEncClick(Sender: TObject);
var StrResult,sRead : String;
    FS : TFileStream;
begin
  memFileTersisip.Lines.Clear;
  if (edPassEnc.Text <> '') and (Length(EdPassEnc.Text) = 16)
```

```
 and (edFileSisip.Text <> '') and (edCarrEnc.Text <> '')then
begin
  if SizeFileSisip <= SizeMedium then
   begin
    if ChkCompr.Checked then
     begin
       zf1.FileName := Zipped;
       zf1.OpenArchive(fmCreate);
       zf1.AddFiles(edFileSisip.Text);
       zf1.CloseArchive;
     end
    else
     CopyFile(PChar(edFileSisip.Text),PChar(Zipped),False);
    FS := TFileStream.Create(Zipped,fmOpenRead);
    SetLength(sRead,FS.Size);
    FS.Read(sRead[1],FS.Size);
    FS.Free;
    SetLength(StrResult,SizeFileSisip);
    EncDec(EdPassEnc.Text,sRead,StrResult,0);
    FS := TFileStream.Create(Zipped,fmCreate);
    FS.Write(StrResult[1],SizeFileSisip);
    FS.Free;
    Sisip(StrResult,Encoded);
    memFileTersisip.Lines.LoadFromFile(Encoded);
    btnSaveEnc.Enabled := True;
    MessageDlg('Proses selesai',mtInformation,[mbOk],0);
   end
  else
   MessageDlg('Daya tampung medium tidak mencukupi!!',mtError,[mbOk],0);
 end
else
```

```
   MessageDlg('Lengkapi inputan : File Carrier, Password,
Teks',mtWarning,[mbOk],0);
end;

procedure TfrmMain.btnSaveEncClick(Sender: TObject);
begin
  saved := false;
  sdText.FileName := '';
  if sdText.Execute then
   begin
     memFileTersisip.Lines.SaveToFile(sdText.FileName);
     MessageDlg('File berhasil disimpan..',mtInformation,[mbOk],0);
     sdText.FileName := '';
     saved := true;
   end;
end;

procedure TfrmMain.btnCarrBrwDecClick(Sender: TObject);
begin
  odMedium.FileName := '';
  memCarrDec.Lines.Clear;
  memResult.Lines.Clear;
  memLastResult.Lines.Clear;
  edCarrDec.Clear;
  edPassDec.Clear;
  chkDeCompr.Checked := False;
  btnSaveDec.Enabled := False;
  if odMedium.Execute then
   begin
     edCarrDec.Text := odMedium.FileName;
     odMedium.FileName := '';
     memCarrDec.Lines.LoadFromFile(edCarrDec.Text);
```

```
  end;
end;

procedure TfrmMain.btnProcDecClick(Sender: TObject);
var StrResult : String;
    FS : TFileStream;
    MS : TMemoryStream;
    DecompOK : Boolean;
begin
 memResult.Lines.Clear;
 memLastResult.Lines.Clear;
 if (edCarrDec.Text <> '')and(edPassDec.Text <> '') and
   (Length(edPassDec.Text) = 16) then
  begin
    DecompOK := True;
    memCarrDec.Lines.SaveToFile(Encoded);
    StrResult := Ambil(Encoded);
    if StrResult <> ErrMsg then
     begin
       EncDec(edPassDec.Text,StrResult,StrResult,1);
       FS := TFileStream.Create(Unknown,fmCreate);
       FS.Write(StrResult[1],Length(StrResult));
       FS.Free;
       try
        MS := TMemoryStream.Create;
        MS.LoadFromFile(Unknown);
        if IsTextFile(Unknown) then
         memResult.Lines.LoadFromStream(MS)
        else
         ShowBinary(MS.Memory^, MS.Size, Display, memResult,30);
        MS.Free;
       except
```

```
      MessageDlg('Error dalam menampilkan isi file hasil ekstraksi dari
medium!',
       mtError,[mbOk],0);
     end;
     if chkDecompr.Checked then
      begin
       zf1.FileName := Unknown;
       try
        zf1.OpenArchive;
        FS := TFileStream.Create(Encoded,fmCreate);
        zf1.ExtractToStream('*.*',FS);
        FS.Free;
        zf1.CloseArchive;
       except
        MessageDlg('File error!! Kemungkinan penyebab :'+#13#10+
         '1. File Tidak dikompresi'+#13#10+
         '2. File carrier corrupt'+#13#10+
         '3. Password dekripsi salah',
         mtError,[mbOk],0);
        zf1.CloseArchive;
        DecompOk := False;
       end;
      end
     else
      CopyFile(PChar(Unknown),PChar(Encoded),False);
     if DecompOk then
     try
      MS := TMemoryStream.Create;
      MS.LoadFromFile(Encoded);
      if IsTextFile(Encoded) then
       memLastResult.Lines.LoadFromStream(MS)
      else
```

```
        ShowBinary(MS.Memory^, MS.Size, Display, memLastResult,30);
        MS.Free;
       except
        MessageDlg('Error dalam menampilkan isi file terakhir!',
         mtError,[mbOk],0);
       end;
       MessageDlg('Proses selesai!',mtInformation,[mbOk],0);
       btnSaveDec.Enabled := True;
     end
    else
     begin
      MessageDlg('File carrier kemungkinan belum terisi atau corrupt',
       mtError,[mbOk],0);
     end;
  end
 else
  MessageDlg('Lengkapi inputan : File Carrier, Password,
Teks',mtWarning,[mbOk],0);
end;

procedure TfrmMain.btnSaveDecClick(Sender: TObject);
begin
 sdResult.FileName := '';
 if sdResult.Execute then
  begin
   CopyFile(PChar(encoded),PChar(sdResult.FileName),False);
   MessageDlg('File tersimpan..',mtInformation,[mbOk],0);
  end;
end;

procedure TfrmMain.pcUtamaChanging(Sender: TObject;
 var AllowChange: Boolean);
```

```
var proses,msg:string;
    resu:integer;
begin
  if pcUtama.ActivePageIndex = 0 then
   begin
     if (btnSaveEnc.Enabled) or (btnSaveText.Enabled) then
      begin
        if pcEnc.ActivePageIndex = 0 then
         proses := 'teks'
        else
         proses := 'file';
        msg := 'Simpan '+proses+' hasil proses enkripsi??';
        resu := MessageDlg(msg,mtConfirmation,mbYesNoCancel,0);
        if resu = mrYes then
        if pcEnc.ActivePageIndex = 0 then
         btnSaveTextClick(Self)
        else
         btnSaveEncClick(Self);
        if not(saved) then saved := resu = mrNo;
        AllowChange := not((resu = mrCancel)or(not saved));
      end
   end
  else
   begin
     if btnSaveDec.Enabled then
      begin
        msg := 'Simpan file hasil proses dekripsi??';
        resu := MessageDlg(msg,mtConfirmation,mbYesNoCancel,0);
        if resu = mrYes then btnSaveDecClick(Self);
        if not(saved) then saved := resu = mrNo;
        AllowChange := not((resu = mrCancel)or(not saved));
      end
```

```
  end;
end;

procedure TfrmMain.pcEncChanging(Sender: TObject;
  var AllowChange: Boolean);
var resu:integer;
begin
  if pcEnc.ActivePageIndex = 0 then
   begin
    if btnSaveText.Enabled then
     begin
      resu := MessageDlg('Simpan teks hasil
enkripsi??',mtConfirmation,mbYesNoCancel,0);
      if resu = mrYes then btnSaveTextClick(Self);
      if not(saved) then saved := resu = mrNo;
      AllowChange := not((resu = mrCancel)or(not saved));
     end
   end
  else
   begin
    if btnSaveEnc.Enabled then
     begin
      resu := MessageDlg('Simpan file hasil
enkripsi??',mtConfirmation,mbYesNoCancel,0);
      if resu = mrYes then btnSaveEncClick(Self);
      if not(saved) then saved := resu = mrNo;
      AllowChange := not((resu = mrCancel)or(not saved));
     end
   end
end;

end.
```

//Program Lucifer

```
unit Lucifer;

interface

uses Sysutils;

procedure EncDec(Key,Mess:String;var strOut:string;Direction:byte);

implementation
Const EN = 0;
     DE = 1;
     o : array [0..7]of byte = (7, 6, 2, 1, 5, 0, 3, 4);
     pr : array [0..7]of byte = (2, 5, 4, 0, 3, 1, 7, 6);
     s0 : array [0..15]of byte = (12, 15, 7, 10, 14, 13, 11, 0, 2, 6, 3, 1, 9, 4, 5, 8);
     s1 : array [0..15]of byte = (7, 2, 14, 9, 3, 11, 0, 4, 12, 13, 1, 10, 6, 15, 8, 5);
var
   tKey : array[0..127]of byte;
   m : array[0..127]of byte;
```

//Program prosedur algoritma Lucifer

```
procedure DoLucifer(Direction: byte);
var
 tcbindex,tcbcontrol:integer;
 rnd,hi,lo,h_1,h_0 : integer;
 bit,temp1:integer;
 by,index,v : integer;
 tr : array[0..7]of integer;
begin
  h_0 := 0;
```

```
h_1 := 1;

if Direction = DE then
 tcbcontrol := 8
else
 tcbcontrol := 0;

for rnd := 0 to 15 do
 begin

  if Direction = DE then
   tcbcontrol := (tcbcontrol+1) and $F;

  tcbindex := tcbcontrol;

  for by := 0 to 7 do
   begin
    lo := (m[(h_1*64)+(8*by)+7])*8
        +(m[(h_1*64)+(8*by)+6])*4
        +(m[(h_1*64)+(8*by)+5])*2
        +(m[(h_1*64)+(8*by)+4]);

    hi := (m[(h_1*64)+(8*by)+3])*8
        +(m[(h_1*64)+(8*by)+2])*4
        +(m[(h_1*64)+(8*by)+1])*2
        +(m[(h_1*64)+(8*by)]);

    v := (s0[lo]+16*s1[hi])*(1-tKey[(8*tcbindex)+by])
          +(s0[hi]+16*s1[lo])*tKey[(8*tcbindex)+by];

    for temp1 := 0 to 7 do
     begin
```

```
      tr[temp1] := v and 1;
      v := v shr 1;
    end;

   for bit := 0 to 7 do
    begin
      index := (o[bit]+by) and 7;
      temp1 := m[(h_0*64)+(8*index)+bit]
            +tKey[(8*tcbcontrol)+pr[bit]]
            +tr[pr[bit]];
      m[(h_0*64)+(8*index)+bit] := temp1 and 1;
    end;

   if (by < 7) or (Direction = DE) then
    tcbcontrol := (tcbcontrol+1) and $F;

  end;

 temp1 := h_0;
 h_0 := h_1;
 h_1 := temp1;

end;

for by := 0 to 7 do
 begin
  for bit := 0 to 7 do
   begin
    temp1 := m[(8*by)+bit];
    m[(8*by)+bit] := m[64+(8*by)+bit];
    m[64+(8*by)+bit] := temp1;
   end;
```

```
  end;

end;
```

//Program prosedur untuk menjalankan Enkripsi/Dekripsi Lucifer

```
procedure EncDec(Key, Mess: String; var strOut: string;
 Direction: byte);
var i,j,c:byte;
   count,k : integer;
begin
 StrOut := '';
 for i := 0 to 15 do
  begin
   c := ord(key[i+1]) and $FF;
   for j := 0 to 7 do
    begin
     tKey[(8*i)+j] := c and 1;
     c := c shr 1;
    end;
  end;
 count := 0;
 for k := 1 to length(Mess) do
  begin
   c := ord(Mess[k]);
   if count = 16 then
    begin
     DoLucifer(Direction);
     for count := 0 to 15 do
      begin
       j := 0;
       for i := 7 downto 0 do
        begin
```

```
            j := (j shl 1) + m[(8*count)+i];
           end;
          strOut := strOut + char(j);
         end;
        count := 0;
      end;
     for i := 0 to 7 do
      begin
        m[(8*count)+i] := c and 1;
        c := c shr 1;
      end;
     inc(count);
   end;
  for k := count to 15 do
   begin
     for i := 0 to 7 do
      begin
        m[(8*k)+i] := 0;
      end;
   end;
  DoLucifer(Direction);
  for count := 0 to 15 do
   begin
    j := 0;
     for i := 7 downto 0 do
      begin
        j := (j shl 1) + m[(8*count)+i];
      end;
     strOut := strOut + char(j);
   end;
end;
```

```
end.
```

//Program Visual Semagram

```
unit VisSem;

interface

uses SysUtils, StrUtils, Classes, Windows;

Var Limit : Integer;
   ErrMsg : String;

function HitungKapasitas(S:String):Integer;
procedure Sisip(S,FN:String);
function Ambil(FN:String):String;

implementation

Const Alfabet = ['A'..'Z','a'..'z','0'..'9'];

function Encode(S: string): string;
var i,j:integer;
   t,tmp:string;
   bit:char;
begin
 result := '';
 for i := 1 to (length(s) div 8) do
  begin
    t := MidStr(s,((i-1)*8)+1,8);
    tmp := '';
    bit := '0';
    for j := 1 to 8 do
```

```
      begin
        if t[j] = bit then
         tmp := tmp + ' '
        else
         begin
           bit := t[j];
           tmp := tmp + #9#32
         end;
      end;
     result := Result + tmp;
   end;
end;

function Decode(S: String): String;
var i:integer;
    tmp : string;
    cnt:byte;
    bit,selesai : boolean;
begin
  selesai := false;
  while not selesai do
   begin
     i := 1;
     cnt := 1;
     bit := False;
     while cnt <= 8 do
      begin
        if s[i] = #9 then
         begin
           bit := not(bit);
           inc(i);
           continue;
```

```
        end;
        tmp := tmp + inttostr((integer(bit)));
        inc(cnt);
        inc(i);
      end;
     s := RightStr(s,Length(s)-i+1);
     selesai := Pos(' ',s) = 0;
   end;
   result := tmp;
end;

function BinToInt(Value: string): byte;
var
  i, iValueSize: byte;
begin
  Result := 0;
  iValueSize := Length(Value);
  for i := iValueSize downto 1 do
    if Value[i] = '1' then
     Result := Result + (1 shl (iValueSize - i));
end;

function IntToBin(Value, Digits: byte): string;
var
  i: byte;
begin
  Result := '';
  for i := Digits downto 0 do
    if Value and (1 shl i) <> 0 then
      Result := Result + '1'
    else
      Result := Result + '0';
```

```
end;

function HitungKapasitas(S: String): Integer;
var ls : TStrings;
    i : integer;
begin
  ls := TStringList.Create;
  ls.LoadFromFile(S);
  Result := 0;
  for i := 0 to ls.Count - 1 do
   if Length(ls.Strings[i]) < (Limit - 2) then
    inc(Result,Limit - Length(ls.Strings[i]));
  dec(Result,ls.Count);
  ls.Free;
end;

function StripTrailing(S: String): String;
var i:integer;
begin
  S := ReverseString(S);
  i := 0;
  while not(S[i+1] in Alfabet) do
   inc(i);
  S := ReverseString(S);
  S := LeftStr(S,Length(S) - i);
  Result := S;
end;

function Ambil(FN: String): String;
var ls:TStrings;
  de,t:string;
  j,i:integer;
```

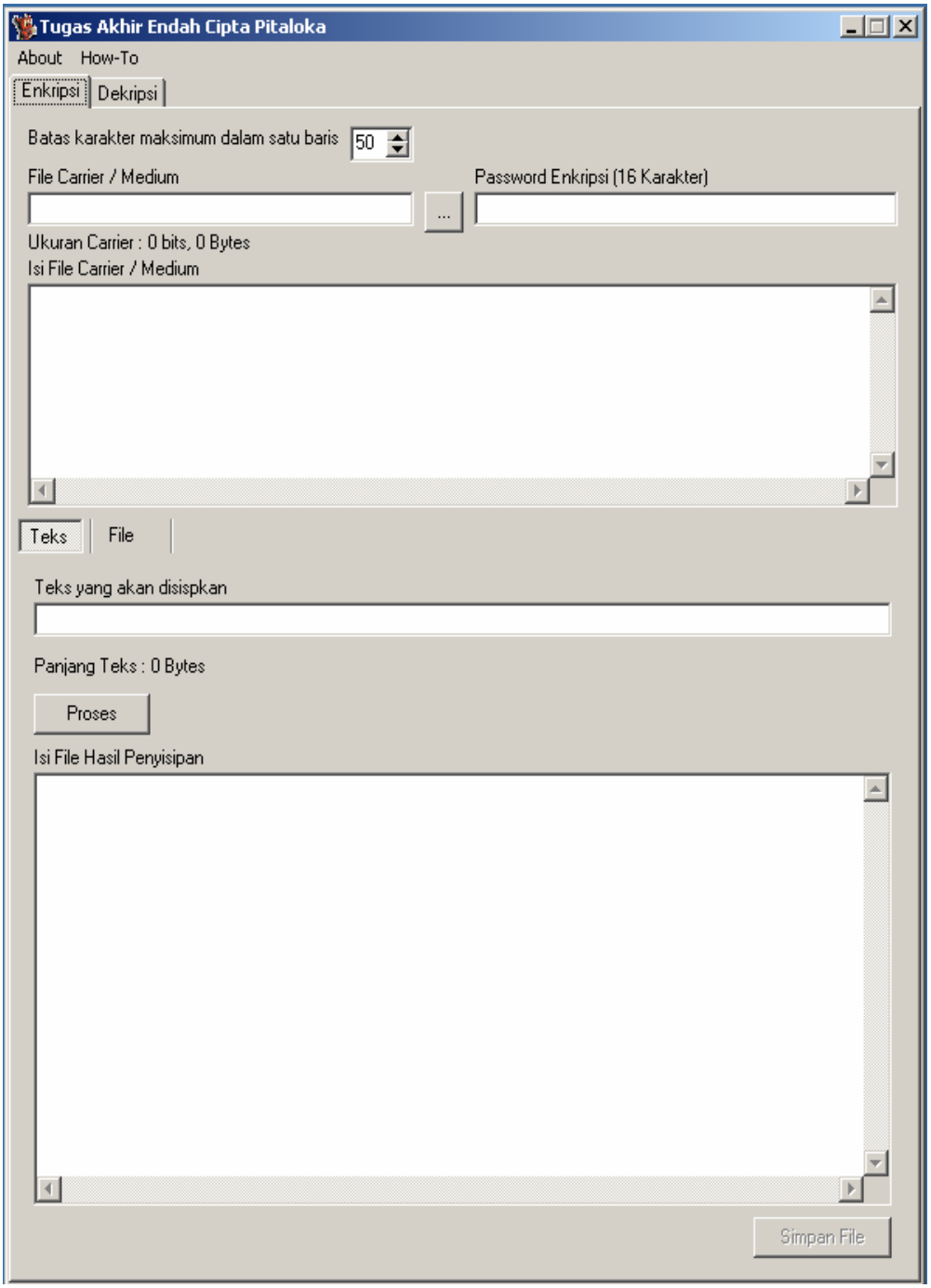
```
  selesai : boolean;
begin
  ErrMsg := IntToStr(GetTickCount);
  ls := TStringList.Create;
  ls.LoadFromFile(FN);
  i := 0;
  selesai := false;
  de := '';
  while not Selesai do
   begin
     t := ls.Strings[i];
     t := ReverseString(t);
     j := 1;
     while(not(t[j] in Alfabet))and(j<>Length(t))do
      inc(j);
     if (j = Length(t)) and (not(t[j] in Alfabet)) then
      begin
        de := de + RightStr(ReverseString(t),j-1);
      end
     else
     if j > 1 then
      begin
        t := ReverseString(t);
        de := de + RightStr(t,j-2);
      end;
     inc(i);
     Selesai := (i = ls.Count);//or(j = 1);
   end;
  ls.Free;
  if de = '' then
   begin
     Result := ErrMsg;
```

```
    exit;
  end;
  de := Decode(de);
  t := '';
  if (Length(de) mod 8) <> 0 then
   t := ErrMsg
  else
    for i := 1 to (Length(de) div 8) do
     t := t + Chr(BinToInt(MidStr(de,((i-1)*8)+1,8)));
  Result := t;
end;

procedure Sisip(S, FN: String);
var ls:TStrings;
    en,t,mess : string;
    j,i,k:Integer;
begin
  ls := TStringList.Create;
  ls.LoadFromFile(FN);
  en := '';
  for i := 1 to Length(s) do
   en := en + IntToBin(Ord(s[i]),7);
  en := encode(en);
  i := 0;
  while en <> '' do
   begin
     t := StripTrailing(ls.Strings[i]);
     j := Length(t);
     if j < (Limit - 2) then
      begin
        j := Limit - j;
        mess := LeftStr(en,j);
```

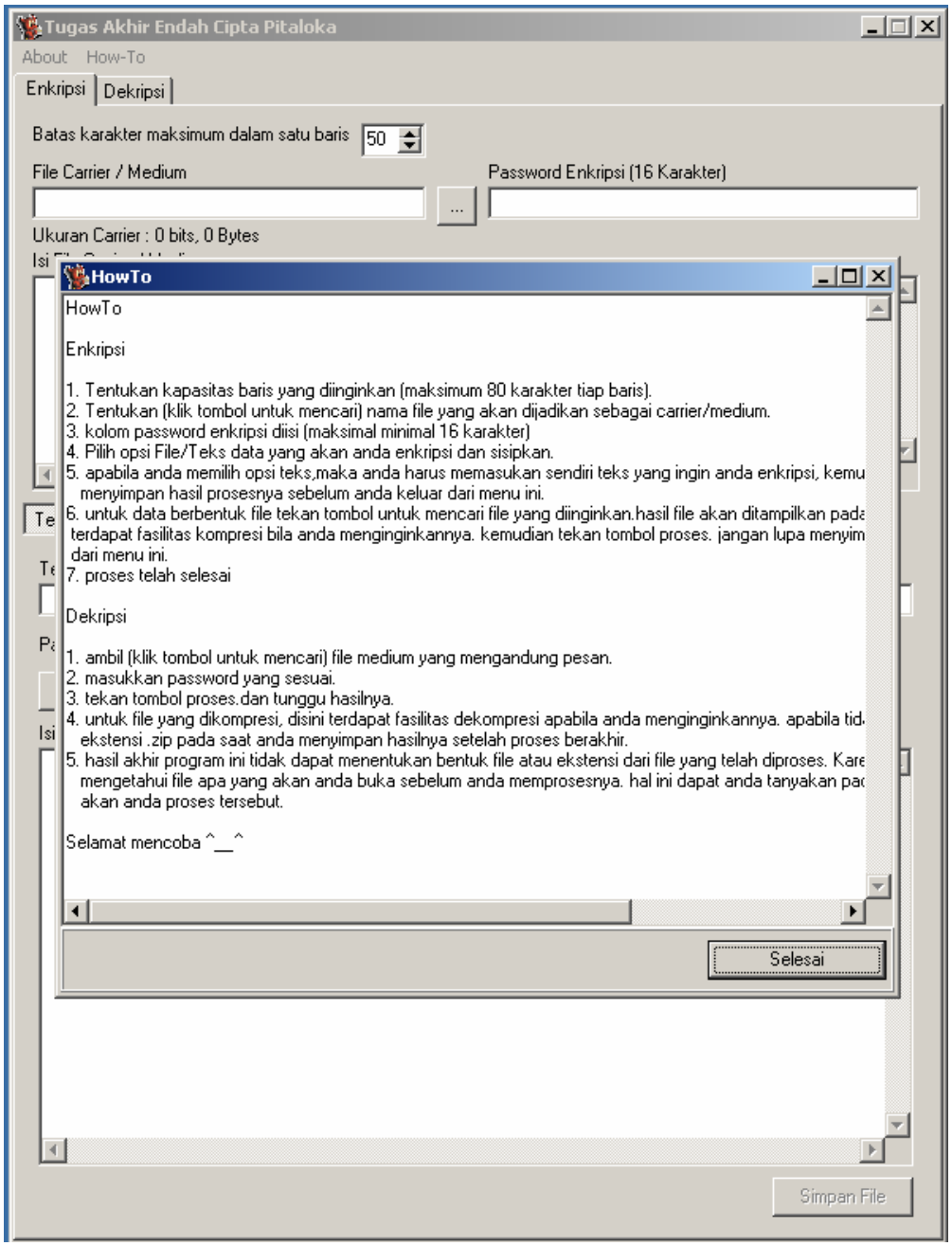
```
        t := t + #9;
        for k := 1 to Length(mess) do
         t := t + en[k];
        ls.Strings[i] := t;
        en := RightStr(en,Length(en) - Length(Mess));
      end;
      inc(i);
   end;
  ls.SaveToFile(FN);
  ls.Free;
end;



end.
```

LAMPIRAN B

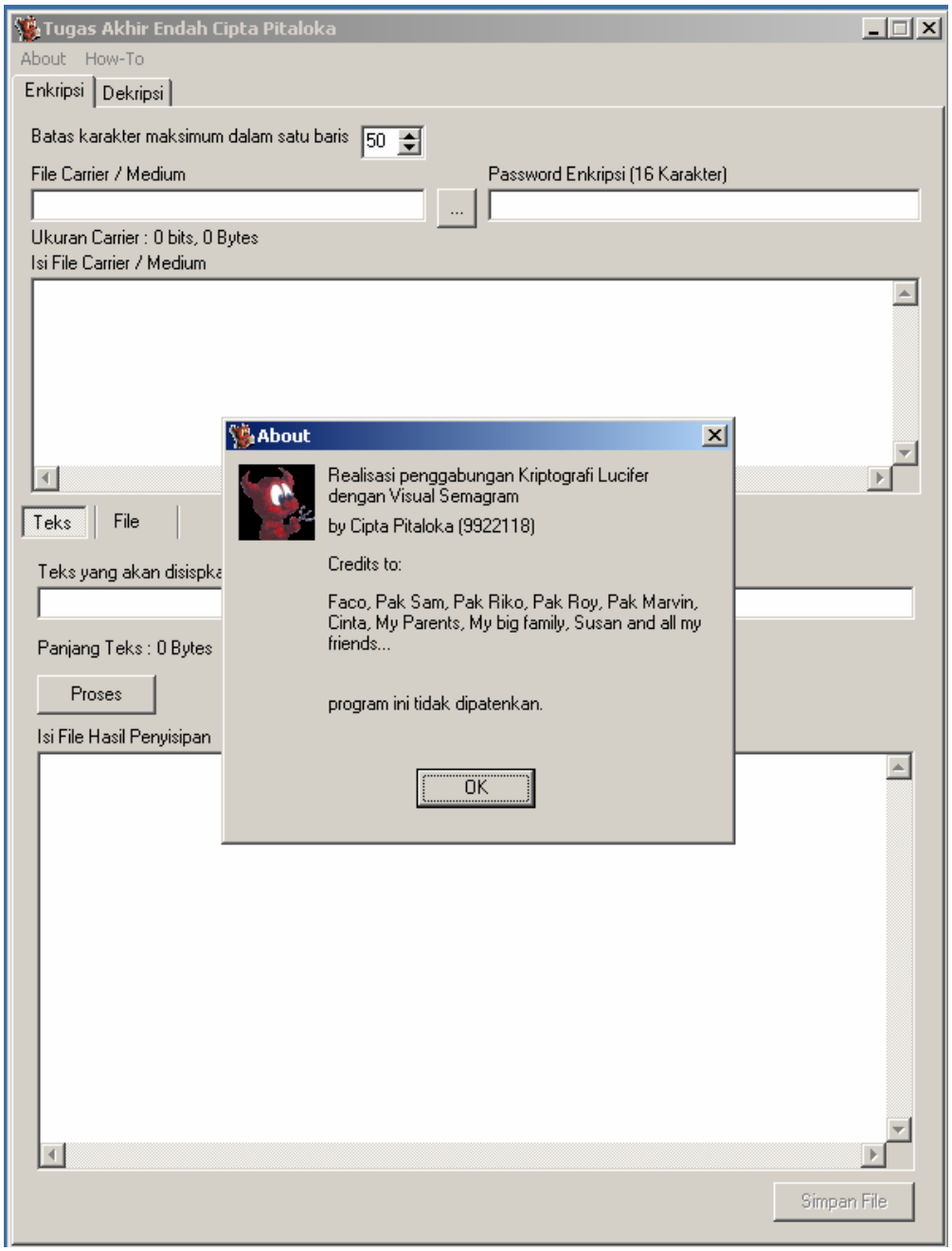
TAMPILAN PROGRAM PENGAMANAN DATA



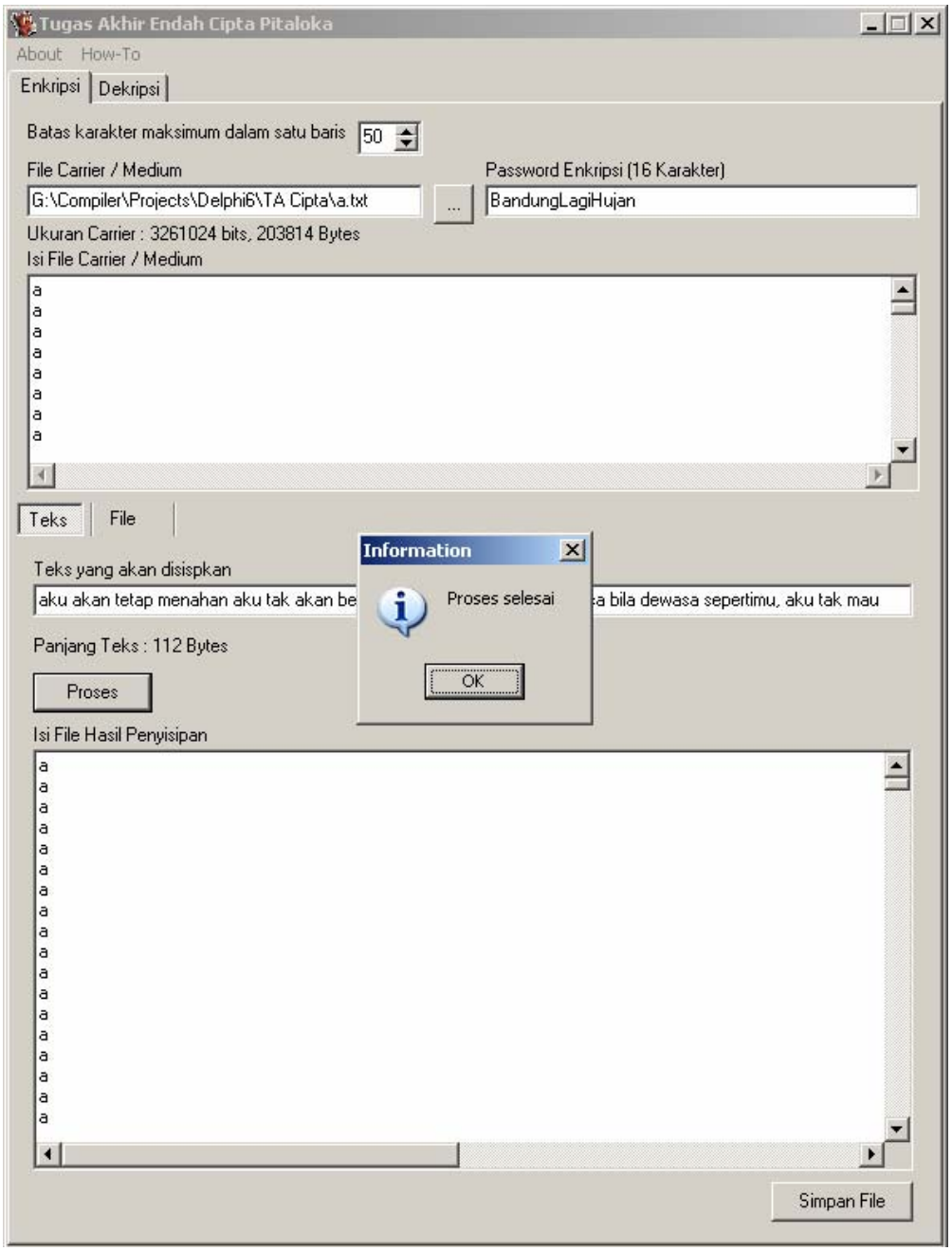
Tampilan Awal Program Pengamanan Data



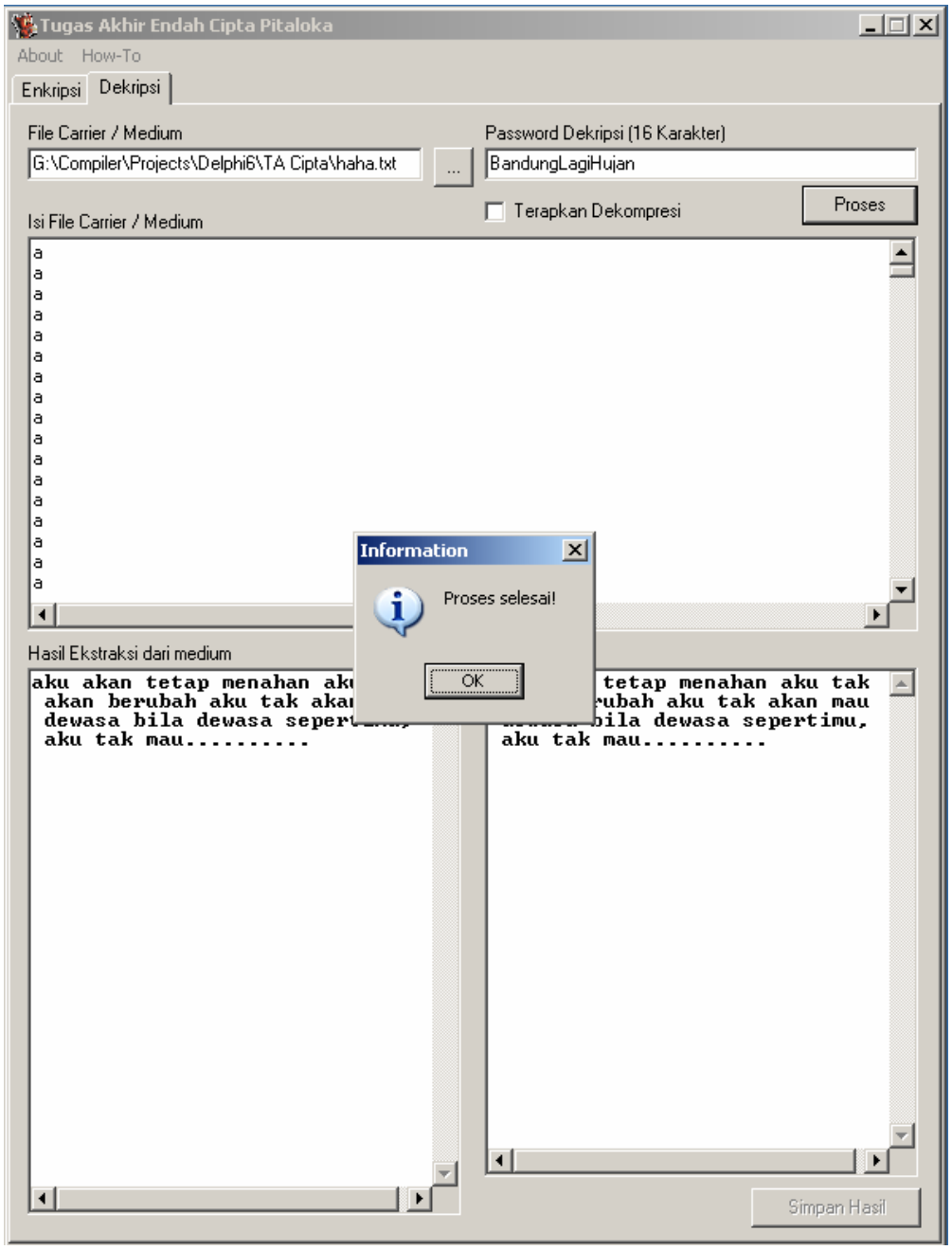
Tampilan Menu How To



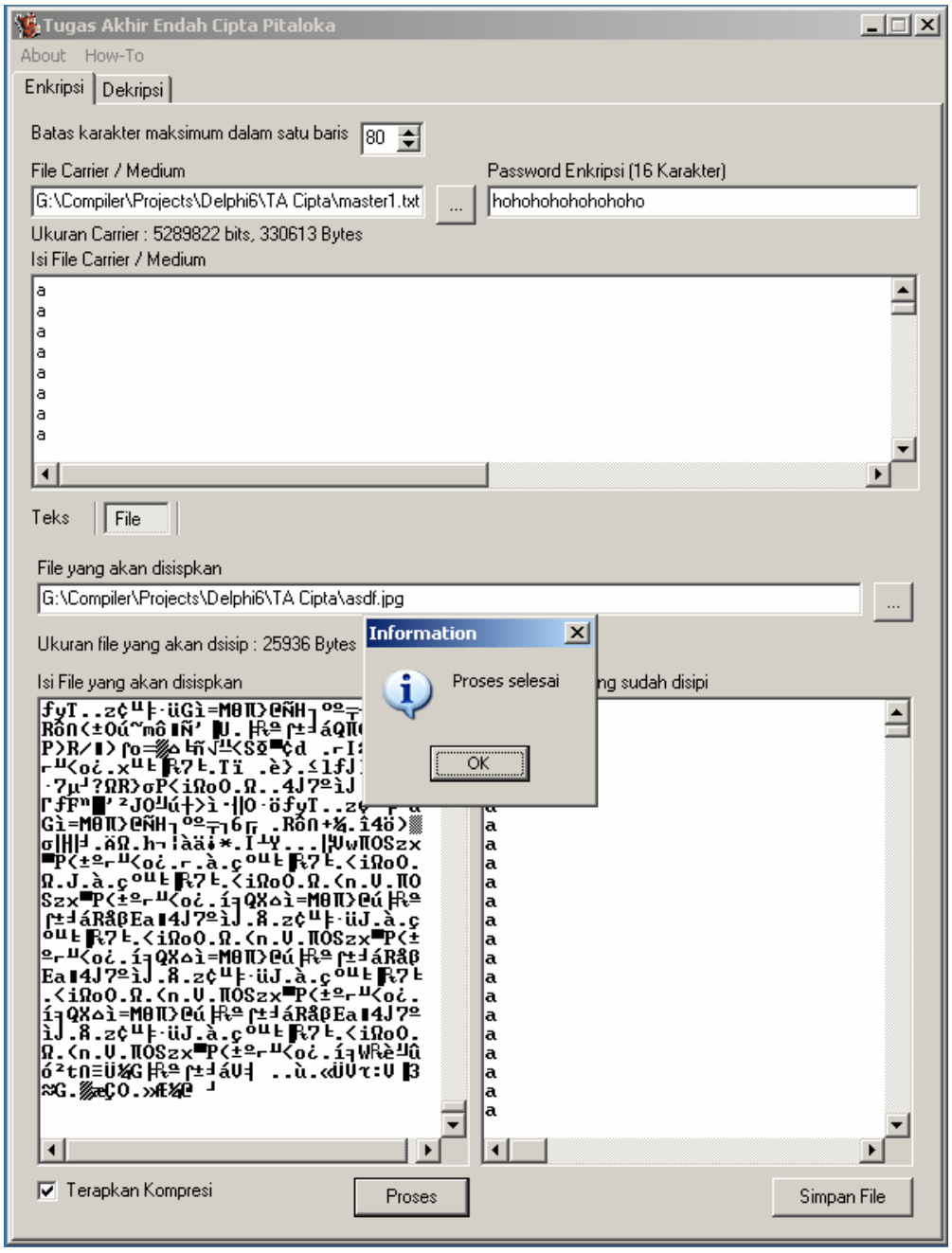
Tampilan Menu About



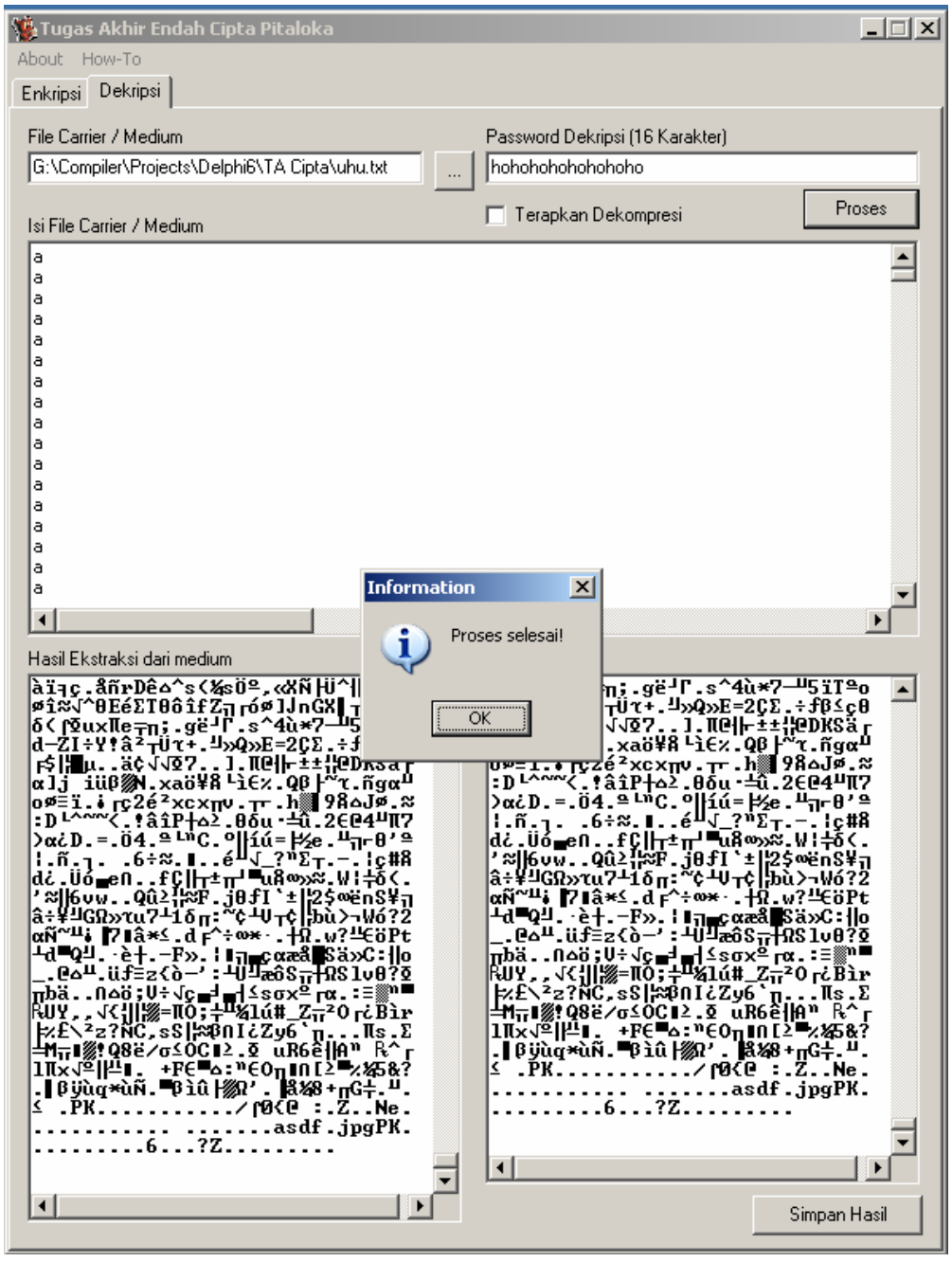
Tampilan Menu Program Enkripsi Teks



Tampilan Menu Dekripsi Teks



Tampilan Menu Enkripsi File



Menu Tampilan Dekripsi File

LAMPIRAN C
TABEL KONFERSI VISUAL SEMAGRAM

Tabel konversi pada algoritma Visual Semagram

Desimal	Karakter	Binari	Visual Semagram	Panjang
0	NULL	0	SSSSSSSS	8
1		1	SSSSSSSTS	9
3		11	SSSSSSTSS	9
7		111	SSSSSTSSS	9
15	□	1111	SSSSTSSSS	9
31		11111	SSSTSSSSS	9
63	?	111111	SSTSSSSSS	9
127	□	1111111	STSSSSSSS	9
255	ÿ	11111111	TSSSSSSSS	9
2		10	SSSSSSTSTS	10
4		100	SSSSSTSTSS	10
6		110	SSSSSTSSTS	10
8		1000	SSSSTSTSSS	10
12		1100	SSSSTSSTSS	10
14		1110	SSSSTSSSTS	10
16	□	10000	SSSTSTSSSS	10
24	□	11000	SSSTSSTSSS	10
28	□	11100	SSSTSSSTSS	10
30	-	11110	SSSTSSSSTS	10
32		100000	SSTSTSSSSS	10
48	0	110000	SSTSSTSSSS	10
56	8	111000	SSTSSSTSSS	10
60	<	111100	SSTSSSSTSS	10
62	>	111110	SSTSSSSSTS	10
64	@	1000000	STSTSSSSSS	10
96	`	1100000	STSSTSSSSS	10
112	p	1110000	STSSSTSSSS	10
120	x	1111000	STSSSSTSSS	10
124	|	1111100	STSSSSSTSS	10

126	~	1111110	STSSSSSSTS	10
128	€	10000000	TSTSSSSSSS	10
192	À	11000000	TSSTSSSSSS	10
224	à	11100000	TSSSTSSSSS	10
240	ð	11110000	TSSSSTSSSS	10
248	ø	11111000	TSSSSSTSSS	10
252	ü	11111100	TSSSSSSTSS	10
254	þ	11111110	TSSSSSSSTS	10
5		101	SSSSSTSTSTS	11
9		1001	SSSSTSTSSTS	11
11		1011	SSSSTSTSTSS	11
13		1101	SSSSTSSTSTS	11
17	□	10001	SSSTSTSSSTS	11
19	□	10011	SSSTSTSSTSS	11
23	□	10111	SSSTSTSTSSS	11
25	□	11001	SSSTSSTSSTS	11
27	□	11011	SSSTSSTSTSS	11
29	□	11101	SSSTSSSTSTS	11
33	!	100001	SSTSTSSSSTS	11
35	#	100011	SSTSTSSSTSS	11
39	'	100111	SSTSTSSTSSS	11
47	/	101111	SSTSTSTSSSS	11
49	1	110001	SSTSSTSSSTS	11
51	3	110011	SSTSSTSSTSS	11
55	7	110111	SSTSSTSTSSS	11
57	9	111001	SSTSSSTSSTS	11
59	;	111011	SSTSSSTSTSS	11
61	=	111101	SSTSSSSTSTS	11
65	A	1000001	STSTSSSSSTS	11
67	C	1000011	STSTSSSSTSS	11
71	G	1000111	STSTSSSTSSS	11
79	O	1001111	STSTSSTSSSS	11
95	_	1011111	STSTSTSSSSS	11
97	a	1100001	STSSTSSSSTS	11
99	c	1100011	STSSTSSSTSS	11
103	g	1100111	STSSTSSTSSS	11
111	o	1101111	STSSTSTSSSS	11

113	q	1110001	STSSSTSSSTS	11
115	s	1110011	STSSSTSSTSS	11
119	w	1110111	STSSSTSTSSS	11
121	y	1111001	STSSSSTSSTS	11
123	{	1111011	STSSSSTSTSS	11
125	}	1111101	STSSSSSTSTS	11
129	□	10000001	TSTSSSSSSTS	11
131	ƒ	10000011	TSTSSSSSTSS	11
135	‡	10000111	TSTSSSSTSSS	11
143	□	10001111	TSTSSSTSSSS	11
159	Ÿ	10011111	TSTSSTSSSSS	11
191	¿	10111111	TSTSTSSSSSS	11
193	Á	11000001	TSSTSSSSSTS	11
195	Ã	11000011	TSSTSSSSTSS	11
199	Ç	11000111	TSSTSSSTSSS	11
207	Ï	11001111	TSSTSSTSSSS	11
223	ß	11011111	TSSTSTSSSSS	11
225	á	11100001	TSSSTSSSSTS	11
227	ã	11100011	TSSSTSSSTSS	11
231	ç	11100111	TSSSTSSTSSS	11
239	ï	11101111	TSSSTSTSSSS	11
241	ñ	11110001	TSSSSTSSSTS	11
243	ó	11110011	TSSSSTSSTSS	11
247	÷	11110111	TSSSSTSTSSS	11
249	ù	11111001	TSSSSSTSSTS	11
251	û	11111011	TSSSSSTSTSS	11
253	ý	11111101	TSSSSSSTSTS	11
10		1010	SSSSTSTSTSTS	12
18	□	10010	SSSTSTSSTSTS	12
20	□	10100	SSSTSTSTSTSS	12
22	□	10110	SSSTSTSTSSTS	12
26	□	11010	SSSTSSTSTSTS	12
34	"	100010	SSTSTSSSTSTS	12
36	$	100100	SSTSTSSTSTSS	12
38	&	100110	SSTSTSSTSSTS	12
40	(	101000	SSTSTSTSTSSS	12
44	,	101100	SSTSTSTSSTSS	12

46	.	101110	SSTSTSTSSSTS	12
50	2	110010	SSTSSTSSTSTS	12
52	4	110100	SSTSSTSTSTSS	12
54	6	110110	SSTSSTSTSSTS	12
58	:	111010	SSTSSSTSTSTS	12
66	B	1000010	STSTSSSSTSTS	12
68	D	1000100	STSTSSSTSTSS	12
70	F	1000110	STSTSSSTSSTS	12
72	H	1001000	STSTSSTSTSSS	12
76	L	1001100	STSTSSTSSTSS	12
78	N	1001110	STSTSSTSSSTS	12
80	P	1010000	STSTSTSTSSSS	12
88	X	1011000	STSTSTSSTSSS	12
92	\	1011100	STSTSTSSSTSS	12
94	^	1011110	STSTSTSSSSTS	12
98	b	1100010	STSSTSSSTSTS	12
100	d	1100100	STSSTSSTSTSS	12
102	f	1100110	STSSTSSTSSTS	12
104	h	1101000	STSSTSTSTSSS	12
108	l	1101100	STSSTSTSSTSS	12
110	n	1101110	STSSTSTSSSTS	12
114	r	1110010	STSSSTSSTSTS	12
116	t	1110100	STSSSTSTSTSS	12
118	v	1110110	STSSSTSTSSTS	12
122	z	1111010	STSSSSTSTSTS	12
130	‚	10000010	TSTSSSSSTSTS	12
132	„	10000100	TSTSSSSTSTSS	12
134	†	10000110	TSTSSSSTSSTS	12
136	ˆ	10001000	TSTSSSTSTSSS	12
140	Œ	10001100	TSTSSSTSSTSS	12
142	Ž	10001110	TSTSSSTSSSTS	12
144	□	10010000	TSTSSTSTSSSS	12
152	˜	10011000	TSTSSTSSTSSS	12
156	œ	10011100	TSTSSTSSSTSS	12
158	ž	10011110	TSTSSTSSSSTS	12
160		10100000	TSTSTSTSSSSS	12
176	°	10110000	TSTSTSSTSSSS	12

184	¸	10111000	TSTSTSSSTSSS	12
188	¼	10111100	TSTSTSSSSTSS	12
190	¾	10111110	TSTSTSSSSSTS	12
194	Â	11000010	TSSTSSSSTSTS	12
196	Ä	11000100	TSSTSSSTSTSS	12
198	Æ	11000110	TSSTSSSTSSTS	12
200	È	11001000	TSSTSSTSTSSS	12
204	Ì	11001100	TSSTSSTSSTSS	12
206	Î	11001110	TSSTSSTSSSTS	12
208	Ð	11010000	TSSTSTSTSSSS	12
216	Ø	11011000	TSSTSTSSTSSS	12
220	Ü	11011100	TSSTSTSSSTSS	12
222	Þ	11011110	TSSTSTSSSSTS	12
226	â	11100010	TSSSTSSSTSTS	12
228	ä	11100100	TSSSTSSTSTSS	12
230	æ	11100110	TSSSTSSTSSTS	12
232	è	11101000	TSSSTSTSTSSS	12
236	ì	11101100	TSSSTSTSSTSS	12
238	î	11101110	TSSSTSTSSSTS	12
242	ò	11110010	TSSSSTSSTSTS	12
244	ô	11110100	TSSSSTSTSTSS	12
246	ö	11110110	TSSSSTSTSSTS	12
250	ú	11111010	TSSSSSTSTSTS	12
21	□	10101	SSSTSTSTSTSTS	13
37	%	100101	SSTSTSSTSTSTS	13
41	)	101001	SSTSTSTSTSSTS	13
43	+	101011	SSTSTSTSTSTSS	13
45	-	101101	SSTSTSTSSTSTS	13
53	5	110101	SSTSSTSTSTSTS	13
69	E	1000101	STSTSSSTSTSTS	13
73	I	1001001	STSTSSTSTSSTS	13
75	K	1001011	STSTSSTSTSTSS	13
77	M	1001101	STSTSSTSSTSTS	13
81	Q	1010001	STSTSTSTSSSTS	13
83	S	1010011	STSTSTSTSSTSS	13
87	W	1010111	STSTSTSTSTSSS	13
89	Y	1011001	STSTSTSSTSSTS	13

91	[	1011011	STSTSTSSTSTSS	13
93	]	1011101	STSTSTSSSTSTS	13
101	e	1100101	STSSTSSTSTSTS	13
105	i	1101001	STSSTSTSTSSTS	13
107	k	1101011	STSSTSTSTSTSS	13
109	m	1101101	STSSTSTSSTSTS	13
117	u	1110101	STSSSTSTSTSTS	13
133	…	10000101	TSTSSSSTSTSTS	13
137	‰	10001001	TSTSSSTSTSSTS	13
139	‹	10001011	TSTSSSTSTSTSS	13
141	□	10001101	TSTSSSTSSTSTS	13
145	‘	10010001	TSTSSTSTSSSTS	13
147	“	10010011	TSTSSTSTSSTSS	13
151	—	10010111	TSTSSTSTSTSSS	13
153	™	10011001	TSTSSTSSTSSTS	13
155	›	10011011	TSTSSTSSTSTSS	13
157	□	10011101	TSTSSTSSSTSTS	13
161	¡	10100001	TSTSTSTSSSSTS	13
163	£	10100011	TSTSTSTSSSTSS	13
167	§	10100111	TSTSTSTSSTSSS	13
175	¯	10101111	TSTSTSTSTSSSS	13
177	±	10110001	TSTSTSSTSSSTS	13
179	³	10110011	TSTSTSSTSSTSS	13
183	·	10110111	TSTSTSSTSTSSS	13
185	¹	10111001	TSTSTSSSTSSTS	13
187	»	10111011	TSTSTSSSTSTSS	13
189	½	10111101	TSTSTSSSSTSTS	13
197	Å	11000101	TSSTSSSTSTSTS	13
201	É	11001001	TSSTSSTSTSSTS	13
203	Ë	11001011	TSSTSSTSTSTSS	13
205	Í	11001101	TSSTSSTSSTSTS	13
209	Ñ	11010001	TSSTSTSTSSSTS	13
211	Ó	11010011	TSSTSTSTSSTSS	13
215	×	11010111	TSSTSTSTSTSSS	13
217	Ù	11011001	TSSTSTSSTSSTS	13
219	Û	11011011	TSSTSTSSTSTSS	13
221	Ý	11011101	TSSTSTSSSTSTS	13

229	å	11100101	TSSSTSSTSTSTS	13
233	é	11101001	TSSSTSTSTSSTS	13
235	ë	11101011	TSSSTSTSTSTSS	13
237	í	11101101	TSSSTSTSSTSTS	13
245	õ	11110101	TSSSSTSTSTSTS	13
42	*	101010	SSTSTSTSTSTSTS	14
74	J	1001010	STSTSSTSTSTSTS	14
82	R	1010010	STSTSTSTSSTSTS	14
84	T	1010100	STSTSTSTSTSTSS	14
86	V	1010110	STSTSTSTSTSSTS	14
90	Z	1011010	STSTSTSSTSTSTS	14
106	j	1101010	STSSTSTSTSTSTS	14
138	Š	10001010	TSTSSSTSTSTSTS	14
146	’	10010010	TSTSSTSTSSTSTS	14
148	”	10010100	TSTSSTSTSTSTSS	14
150	–	10010110	TSTSSTSTSTSSTS	14
154	š	10011010	TSTSSTSSTSTSTS	14
162	¢	10100010	TSTSTSTSSSTSTS	14
164	¤	10100100	TSTSTSTSSTSTSS	14
166	¦	10100110	TSTSTSTSSTSSTS	14
168	¨	10101000	TSTSTSTSTSTSSS	14
172	¬	10101100	TSTSTSTSTSSTSS	14
174	®	10101110	TSTSTSTSTSSSTS	14
178	²	10110010	TSTSTSSTSSTSTS	14
180	´	10110100	TSTSTSSTSTSTSS	14
182	¶	10110110	TSTSTSSTSTSSTS	14
186	º	10111010	TSTSTSSSTSTSTS	14
202	Ê	11001010	TSSTSSTSTSTSTS	14
210	Ò	11010010	TSSTSTSTSSTSTS	14
212	Ô	11010100	TSSTSTSTSTSTSS	14
214	Ö	11010110	TSSTSTSTSTSSTS	14
218	Ú	11011010	TSSTSTSSTSTSTS	14
234	ê	11101010	TSSSTSTSTSTSTS	14
85	U	1010101	STSTSTSTSTSTSTS	15
149	•	10010101	TSTSSTSTSTSTSTS	15
165	¥	10100101	TSTSTSTSSTSTSTS	15
169	©	10101001	TSTSTSTSTSTSSTS	15

171	«	10101011	TSTSTSTSTSTSTSS	15
173		10101101	TSTSTSTSTSSTSTS	15
181	µ	10110101	TSTSTSSTSTSTSTS	15
213	Õ	11010101	TSSTSTSTSTSTSTS	15
170	ª	10101010	TSTSTSTSTSTSTSTS	16