ANNEXURE E: SOLUTION FOR QUESTION 2: DELPHI

QUEST2 CLASS UNIT

```
unit uQuest2_Memo;
 {*** Solution for class unit of question 2 ***}
interface

TYPE
   TQuest2 = class(TObject)
     private
       fAType    : String;
       fNumber   : Integer;
       fSize     : Real;
       fCat      : Char;
     public
       constructor create(sAType: String;iNum: integer;rSize: Real;cCat: Char);
       function toString:String;
       function isSuitable(cCat:char; iNumber:integer):Boolean;
       procedure setAType(sAType : String);
       procedure setNumber(iNumber : Integer);
       procedure setSize(rSize : Real);
       procedure setCat(cCat : Char);
       function getAType:String;
       function getNumber:integer;
       function getSize:real;
       function getCat:Char;
   end;

implementation

uses SysUtils;

{ TQuest2 }

constructor TQuest2.create(sAType: String;iNum: integer;rSize: Real;cCat:
Char);
begin
   fAType  := sAType;
   fNumber := iNum;
   fSize  := rSize;
   fCat    := cCat;
end;

function TQuest2.isSuitable(cCat:char; iNumber:integer):Boolean;
var
  rSpace :real;
begin
    Result := false;
    if fAType = 'XXX' then
    begin
        rSpace := fSize / iNumber;
        case cCat of
        'L': Result := rSpace >= 18;
        'M': Result := (rSpace >= 12) and (rSpace < 18);
        'S' : Result := (rSpace >= 7) and (rSpace < 12);
        end;
     end;
end;

function TQuest2.toString:String;
begin
```

```
  Result := fAType  + '...' + fCat + #13   + 'Enclosure size: ' +
      FloatToStrF(fSize, ffFixed, 8,1) + #13  +'Number of animals: ' +
      IntToStr(fNumber) +#13 + #13;
end;

procedure TQuest2.setAType(sAType: String);
begin
   fAType := sAType;
end;

procedure TQuest2.setSize(rSize: Real);
begin
   fSize := rSize;
end;

procedure TQuest2.setCat(cCat: Char);
begin
   fCat := cCat;
end;

procedure TQuest2.setNumber(iNumber: Integer);
begin
   fNumber := iNumber;
end;

function TQuest2.getAType:String;
begin
   Result := fAType;
end;

function TQuest2.getNumber:integer;
begin
   Result := fNumber;
end;

function TQuest2.getSize:real;
begin
   Result := fSize;
end;

function TQuest2.getCat:Char;
begin
   Result := fCat;
end;

end.
```

MAIN FORM UNIT

```
unit Question2U_Memo;
  {*** Solution for main unit of question 2 ***}

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, Menus,
  uQuest2_Memo;

type
  TfrmQ2 = class(TForm)
    mnuMain: TMainMenu;
    mnuOptionA: TMenuItem;
    mnuQuit: TMenuItem;
    redQ2: TRichEdit;
    mnuOptionB: TMenuItem;
    procedure mnuQuitClick(Sender: TObject);
    procedure mnuOptionbClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure mnuOptionAClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQ2: TfrmQ2;

implementation

var
  EnclosuresArr :array[1..30] of TQuest2;
  iCount :integer;

{$R *.dfm}
{$R+}

procedure TfrmQ2.FormCreate(Sender: TObject);
var
   TFile : TextFile;
   iPos, iNumber : integer;
   rSize :real;
   cCat  :Char;
   sLine, sAnimal :String;
begin
   if FileExists ('DataQ2.txt') <> true    then
     begin
       ShowMessage('File does not exist');
       Exit;
     end;
   AssignFile(TFile, 'DataQ2.txt');
   Reset(TFile);

   iCount := 0;
   while NOT EOF(TFile) AND (iCount < 30) do
   begin
     inc(iCount);
     readln(TFile, sLine);
     iPos := pos(';', sLine);
     sAnimal := copy(sLine, 1, iPos -1);
     delete(sLine, 1, iPos);

     iPos := pos('#', sLine);
     iNumber := StrToInt(copy(sLine, 1, iPos -1));
     delete(sLine, 1, iPos);

     iPos := pos(';', sLine);
     rSize := StrToFloat(copy(sLine, 1, iPos -1));
     delete(sLine, 1, iPos);

     cCat := sLine[1];

     EnclosuresArr[iCount] := TQuest2.create(sAnimal, iNumber, rSize, cCat);
   end;
   closeFile(TFile);
```

```
end;

procedure TfrmQ2.mnuOptionAClick(Sender: TObject);
var
 K :integer;
begin
     redQ2.Lines.Add('List of all enclosures');
     redQ2.Lines.Add('===================');
     For K := 1 to iCount do
       begin
          redQ2.Lines.Add('Enclosure number: ' + IntToStr(K) + #13 +
EnclosuresArr[K].toString);
     end;
end;

procedure TfrmQ2.mnuOptionBClick(Sender: TObject);
var
  K,iNum :integer;
  bFound :boolean;
  cCat   :char;
  sAType :String;
begin
  sAType := InputBox('Animal type', 'Enter the type of animal for example
Tiger','Tiger');
  iNum := StrToInt(InputBox('Number of animals', 'Enter the number of
animals','2'));
  cCat := InputBox('Category', 'Enter the category (L/M/S)','L')[1];
  bFound := false;
  K := 1;
  While (bFound <> true) and (K <= iCount) do
   begin
    if  EnclosuresArr[K].isSuitable(cCat, iNum)  then
       begin
         EnclosuresArr[K].setAType(sAType);
         EnclosuresArr[K].setCat(cCat);
         EnclosuresArr[K].setNumber(iNum);
         bFound := true;
       end
     else
      inc(K);
    end;
    redQ2.Lines.Clear;
    if NOT(bFound) then
      redQ2.Lines.Add('No suitable enclosure was found')
    else
      begin
         redQ2.Lines.Clear;
         redQ2.Lines.Add('These animals were placed in enclosure number ' +
IntToStr(K));
         redQ2.Lines.Add(' ');
         mnuOptionA.Click;
      end;
end;
procedure TfrmQ2.mnuQuitClick(Sender: TObject);
begin
  Application.Terminate;
end;
end.
```