NATIONAL SENIOR CERTIFICATE EXAMINATION
NOVEMBER 2021

INFORMATION TECHNOLOGY: PAPER I

MARKING GUIDELINES

Time: 3 hours 150 marks

These marking guidelines are prepared for use by examiners and sub-examiners, all of whom are required to attend a standardisation meeting to ensure that the guidelines are consistently interpreted and applied in the marking of candidates' scripts.

The IEB will not enter into any discussions or correspondence about any marking guidelines. It is acknowledged that there may be different views about some matters of emphasis or detail in the guidelines. It is also recognised that, without the benefit of attendance at a standardisation meeting, there may be different interpretations of the application of the marking guidelines.

SECTION A

QUESTION 1

Question 1.1 (4)

```
SELECT *
FROM tblDomains
WHERE VPN = TRUE
ORDER BY DomainName
```

Question 1.2 (5)

```
SELECT *
FROM tblServiceAgents
WHERE Experience BETWEEN 2 AND 5 AND
Department IN ("Maintenance" , "Admin")

Alternative
SELECT *
FROM tblServiceAgents
WHERE Experience BETWEEN 2 AND 5 AND
(Department = “Maintenance” or Department = “Admin”)
```

Question 1.3 (4)

```
SELECT COUNT(*) AS RSACompanies
FROM tblDomains
WHERE DomainName LIKE "*.co.za" accept % for mysql / javadb
```

Question 1.4 (7)

```
SELECT DomainName, Package, Cost * 1.5 AS BargainPrice
FROM tblDomains, tblPackages
WHERE MONTH(DateSubscribed) = MONTH(NOW() ) AND Package <> 'platinum'
AND tblDomains.PackageID = tblPackages.PackageID
```

MySQL/JavaDB

```
SELECT DomainName, Package, Cost * 1.5 AS BargainPrice
FROM tblDomains, tblPackages
WHERE MONTH(DateSubscribed) = MONTH(Current_Date)  AND Package <>
'platinum' AND tblDomains.PackageID = tblPackages.PackageID
```

Question 1.5 (7)

```
SELECT Department, AVG (Experience) as AvgExperience
FROM tblServiceAgents
GROUP BY Department
HAVING Avg(Experience)> 6
```

Question 1.6 (6)

```
UPDATE tblDomains
SET DomainName = LEFT (DomainName, LEN(DomainName) – 3) & ".co.ind"
WHERE DomainName LIKE "*.in"
```

Question 1.7 (8)

```
SELECT DomainName, Firstname, Lastname, tblTickets.AgentID,
PriorityLevel, DateLogged
FROM tblDomains, tblServiceAgents, tblTickets
WHERE tblDomains.DomainID = tblTickets.DomainID AND
tblServiceAgents.AgentID = tblTickets.AgentID AND
DateCompleted IS NULL
ORDER BY tblTickets.AgentID, PriorityLevel accept inner joins
```

Question 1.8 (9)

```
INSERT INTO tblDomains (DomainName, DateSubscribed, VPN, PackageID)
SELECT INT (RND(DomainID)  * 5  + 1) & DomainName, NOW() , TRUE,
PackageID(matches to order of insert fields)
FROM tblDomains (Correct insert with select)
WHERE DomainName LIKE "*.ru"
```

MySQL

```
INSERT INTO tblDomains (DomainName, DateSubscribed, VPN, PackageID)
SELECT CONCACT(floor(RAND()  * 5  + 1 ), DomainName ) , CURRENT_TIME ,
TRUE, PackageID (matches to order of insert fields)
FROM tblDomains  (Correct insert with select)
WHERE DomainName LIKE ‘.ru’
```

JavaDB

```
INSERT INTO tblDomains (DomainName, DateSubscribed, VPN, PackageID)
SELECT SUBSTR( '12345' , INTEGER (RANDOM() * 5) + 1 , 1)  || DomainName
, CURRENT_TIME , TRUE, PackageID(matches to order of insert fields)
FROM tblDomains  (Correct insert with select)
WHERE DomainName LIKE ‘%.ru’
```

JAVA SOLUTION

QUESTION 2 TECHNICIAN CLASS

```
//Question 2.1 - 3
public class Technician {  class header

    private String techID;  all fields private
    private String name;    all correctly typed with correct names
    private int experience;
    private String roleSpeciality;

    //Question 2.2 - 4
     correct header
    public Technician(String inTID, String inN, int inE, String inR)
      correct parameter names and types
    {
        techID = inTID;  fields set to parameters
        name = inN;
        experience = inE;
        roleSpeciality = inR;
    }

    //Question 2.3 - 2

     correct header and return type for all four getters
    public String getTechID()
    {
        return techID;
    }


    public String getName()
    {
        return name;
    }

    public int getExperience()
    {
        return experience;
    }

    public String getRoleSpeciality()
    {
        return roleSpeciality;
    }

    //Question 2.4 - 4
     correct header
    public String toString()
    {
         contains all fields
         field in correct format
         return formatted string
        return name +", " + techID + ", " + experience + " year(s) [" +
        roleSpeciality + "]";
    }
}
```

QUESTION 3 SERVER CLASS

```
//Question 3.1 – 5
 class header correct
public class Server {

    private String serverID;     String properties made private
    private String location;     typed correctly
    private String role;         named correctly
    private String fault;

      Technician property named and typed correctly
    private Technician assignedTech;

    //Question 3.2 – 4
      Constant declared with final / constant
      named correctly
      typed correctly
      assigned correct values

    public static final String ROLETYPE_EMAIL = "Email";
    public static final String ROLETYPE_FILE = "File";
    public static final String ROLETYPE_PRINT = "Print";
    public static final String ROLETYPE_CUSTOM = "Custom";

    //Question 3.3 - 6
     constructor named correctly
    public Server (String inSID, String inLo, String inRo , String
inFa)
      parameters correct excluding technician
    {
        server, location and fault assigned correctly
        serverID = inSID;
        location = inLo;
        fault = inFa;

        if statement to check role parameter against Constants
        Check for case sensitivity
         nested correctly assigning default value of custom
        if(inRo.equalsIgnoreCase(ROLETYPE_EMAIL))
        {
            role = inRo;
        }
        else if (inRo.equalsIgnoreCase(ROLETYPE_FILE))
        {
            role = inRo;
        }
        else if (inRo.equalsIgnoreCase(ROLETYPE_PRINT))
        {
            role = inRo;
        }
        else
        {
            role = ROLETYPE_CUSTOM;
        }


    }
    //Question 3.4 - 2
     method headers and returns correct
    public String getServerID() {
        return serverID;
    }


    public String getLocation() {
        return location;
    }


    public String getRole() {
        return role;
    }


    public String getFault() {
        return fault;
    }

    //Question 3.5 - 2

     method header and return correct
    public Technician getAssignedTech() {
        return assignedTech;
    }

     method header correct accepts Technician parameter

    public void setAssignedTech(Technician inTech) {
        assignedTech = inTech;
    }

    //Question 3.6 - 6
     method header correct
    public String toString()
    {

        String r = "";

         fields added to string
         correct format
        r = r + "Server: " + serverID + "(Role: " + role + ")\n";
        r = r + "Fault: " + fault + " @ " + location+ "\n";
        check if there is a technician assigned and appended correctly
        if(assignedTech != null)
        {

            r = r + "Assigned to: " + assignedTech.toString();
         appended correctly

        }
        else
        {
            r = r + "Assigned to: none assigned";
        }

        return build up string
        return r;
    }

}
```

QUESTION 4, 6.1, 7.1 SERVERMANAGER CLASS

```
//Question 4.1 - 1
 correct class header
public class ServerManager {

    //Question 4.2 - 4
     Both properties private
     Server array declared with correct name
     Array size set to 50
    private Server sArr[] = new Server[50];
    size initialized correctly
    private int size = 0;

    //Question 4.3 - 9
     constructor header correct
    public ServerManager()
    {
       try
       {      open the file for reading
           Scanner sc = new Scanner(new File("servers.txt"));

          loop through all the lines
           while(sc.hasNextLine())
           {
               read the next line from the file
               String line = sc.nextLine();

               split the line into the required parts
               String tokens[] = line.split("#");

               String sid = tokens[0];
               String location = tokens[1];
               String fault = tokens[2];
```

```
                String role = tokens[3];

               create server object

                Server s = new Server(sid, location, fault , role);
               add server to array
                sArr[size] = s;
               increment size
                size++;

            }
            sc.close();
        }
        catch(FileNotFoundException e)
        {

            System.out.println("File Missing");  handle exception
        }
    }

    //Question 4.4 - 5
     method header correct and returns String
    public String allServers()
    {
        String r = ""; string initialized

         loop through server array
        for (int i = 0; i < size; i++) {
            append to string with extra blank line
            r = r + sArr[i].toString() + "\n\n";
        }

        return r;
    }

    //Question 4.5 - 5
     method header correct and returns int
    public int countServers(String fault, String roletype)
    {
        int count = 0;
         loop through array
        for (int i = 0; i < size; i++) {
            check if server fault
           role matches parameters ignoring case
            if(sArr[i].getFault().equalsIgnoreCase(fault) &&
sArr[i].getRole().equalsIgnoreCase(roletype))
            {
                count = count + 1;  update count correctly
            }
        }

        return count;
    }
```

```
//Question 6
//Question 6.1 - 11
       header correct
    public void assignTechnicians()
    {
        try
        {
             open file
            Scanner sc = new Scanner(new File("technicians.txt"));

             loop through file
            while(sc.hasNextLine()) {

                 get lines and split
                String line =  sc.nextLine();
                String tokens[] = line.split("#");
                String tid = tokens[0];
                String name = tokens[1];
                int exp = Integer.parseInt(tokens[2]);
                String rs = tokens[3];
                 create technician object
                Technician t = new Technician(tid, name , exp , rs);

                 looping through all the servers
                  using while looping structure
                  checking that the limit of 4 servers per technician is
                not exceeded
                int limit = 0;
                int k = 0;

                while (k < size & limit < 4) {
                  check to see if the server role matches the technician
                role and that no technicians has been assigned

                if(sArr[k].getRole().equalsIgnoreCase(t.getRoleSpecialit
                y()) & sArr[k].getAssignedTech() == null)
                  {
                        assign technician to server using method using
                       server class
                       sArr[k].setAssignedTech(t);
                        increase number of servers assigned
                       limit++;

                  }
                  k++;
                }

            }
        }
        catch(FileNotFoundException e)
        {
            System.out.println("File Missing " + e.getMessage());
        }
    }
    //Question 7
    //Question 7.1 - 17

    private boolean findServer(String loc, String tID)
    {
        loop through servers
        for (int i = 0; i < size; i++)
        {

            check if techID matches server techid
            if (sArr[i].getAssignedTech() != null &&
sArr[i].getAssignedTech().getTechID().equals(tID) &&
sArr[i].getLocation().equals(loc))
            {
                return true;
            }
        }
        return false;
    }



    public String printMap(String techID)
    {

        creating string for the map
        String map = "";

        create date stamp
        correct format
        DateTimeFormatter formatDate =
DateTimeFormatter.ofPattern("YYYY/MM/dd HH:mm:ss");

        add date to map
        map = map + formatDate.format(LocalDateTime.now()) + "\n";

        create and  append column numbers
        for (int i = 1; i <= 15; i++)
        {
            map += "\t" + i;
        }
        map += "\n";

         create loop for row letters
        for (char row = 'A'; row <= 'J'; row++)
        {
            map += row;
            create loop for columns
            for (int col = 1; col <= 15; col++)
            {
                check if a server location is found
                String loc = (row + "" + col);
                if (findServer(loc, techID))
                {
                    map += "\tX"; appending x
                } else
                {
                    map += "\t*"; appending *
                }
            }
            map += "\n";
        }

        try
        {
           create file to write save map data with techID as file name
     PrintWriter out = new PrintWriter(new FileWriter(techID + ".txt"));

            write map data to file
            out.println(map);
            close file
            out.close();

        } catch (Exception e)
        {
            System.out.println("Failed to write to file");
        }
        return map
        return map;
    }
}
```

QUESTION 5, 6.2 & 7.2 SERVERUI CLASS

```
//Question 5
//Question 5.1 - 1
 application class created with main method
public class ServerUI {

    public static void main (String args[])
    {
        //Question 5.2 - 1
         ServerManager object created in appropriate place in the code
         ServerManager sm = new ServerManager();

        //Question 5.3 - 1
         allServers called and displayed correctly
         System.out.println(sm.allServers());

      //Question 5.4 - 3
         countServers called
        using Constant value and  called correctly
         System.out.println("Number of servers with a Temp fault
and Custom Role" + sm.countServers("Temp", Server.ROLETYPE_CUSTOM));

       //Question 6.2 - 2
      assignedTechnicians called correctly and redisplay allServers
         sm.assignTechnicians();
         System.out.println(sm.allServers());
        //Question 7.2 - 2
         printMap method called
        Correct techID used
         System.out.println(sm.printMap("T-D1"));
    }
}
```

DELPHI SOLUTION

QUESTION 2 TECHNICIAN CLASS

```
unit uTechnician;

interface
uses SysUtils;
//Question 2.1 - 3
 class header
type TTechnician = class
  private          all fields private
    techID, name, roleSpeciality : string;
      named correctly with correct type
    experience : integer;
  public
    constructor Create( inTID : string; inN : string; inE :
integer);
    function getTechID : string;
    function getName : string;
    function getExperience : integer;
    function getRoleSpeciality : integer;
    function toString : string;


end;

implementation

{ TTechnician }
 //Question 2.2 - 4
  header correct
constructor TTechnician.Create(inTID, inN: string; inE: integer,
inR : string);
 correct parameter names and  types
begin
  techID := inTID;      fields set to parameters
  name := inN;
  experience := inE;
  roleSpeciality := inR;

end;

//Question 2.3 - 2
 correct header and return type for all four getters
function TTechnician.getExperience: integer;
begin
  Result:= experience;
end;

function TTechnician.getName: string;
begin
  Result:= name;
```

```
end;

function TTechnician.getRoleSpeciality: string;
begin
  Result:= roleSpeciality
end;
function TTechnician.getTechID: string;
begin
  Result := techID
end;

//Question 2.4 - 4
 correct header
function TTechnician.toString: string;
begin
   contains all fields
   field in correct format
   returned correctly
  Result:= name + ',' + techID + ', ' + IntToStr(experience) + '
year(s)[' + roleSpeciality + ']';
end;

end.
```

QUESTION 3 SERVER CLASS

```
unit uServer;

interface
  uses SysUtils, uTechnician;
   //Question 3.1 - 5
   class header correct
  type TServer = class

    private
      String properties made private
      typed correctly
      named correctly
      serverID, location, role, fault: string;
      Technician property named and typed correctly
      assignedTech : TTechnician;

    public
     //Question 3.2 - 4
      const    Constant declared
        ROLETYPE_EMAIL = 'Email';     named correctly
        ROLETYPE_FILE = 'File';       typed correctly
        ROLETYPE_PRINT = 'Print';      values assigned correctly
        ROLETYPE_CUSTOM = 'Custom';

        constructor Create (inSID, inLo, inFa, inRo : string);
        function getServerID() : string;
        function getLocation() : string;
        function getRole() : string;
        function getFault() : string;
        procedure setAssignedTech(inTech : TTechnician);
        function getAssignedTech() : TTechnician;
        function toString() : string;


  end;

implementation

{ TServer }
//Question 3.3 - 6
 header correct
 parameters correct excluding technician
constructor TServer.Create(inSID, inLo, inFa, inRo: string);
//correct parameters excluding Technician
begin

   server, location and fault correctly assigned
  serverID:= inSID;
  location:= inLo;
  fault:= inFa;
       if statement to check role parameter against Constants
       Check for case sensitivity
      nested correctly assigning default value of custom
  if (CompareText(inRo,ROLETYPE_EMAIL) = 0) or
    (CompareText(inRo,ROLETYPE_FILE) = 0)   or
        (CompareText(inRo,ROLETYPE_PRINT) = 0)  then
    begin
      role := inRo;
    end
  else
    begin
         role := ROLETYPE_CUSTOM
    end;

end;


//Question 3.4 - 2
 method headers correct and returns correct
function TServer.getFault: string;
begin
  Result:= fault;
end;

function TServer.getLocation: string;
begin
  Result:= location;
end;

function TServer.getRole: string;
begin
```

```
  Result:= role;
end;

function TServer.getServerID: string;
begin
  Result:= serverID;
end;

//Question 3.5 - 2
 method header and return correct
function TServer.getAssignedTech: TTechnician;
begin
  Result:= assignedTech; //return technician type
end;

 method header correct and assignment correct
procedure TServer.setAssignedTech(inTech: TTechnician);
begin
  assignedTech := inTech; //assigns correctly
end;

//Question 3.6 - 6
 method header correct
function TServer.toString: string;

begin
     appending into Result
     field added to result
     correct format
    Result:= 'Server: ' + serverID + '(Role: ' + role + ')' +
#13#10;
    Result:= Result + 'Fault: ' + fault + ' @ ' + location +
#13#10;

     check if there is a technician and append toString or "none
assigned"
    if(assignedTech <> nil) then
      begin
        Result:= Result + 'Assigned to: ' +
assignedTech.toString();
      end
    else
      begin
        Result:= Result + 'Assigned to: none assigned';
      end;

      Result built up correctly
end;

end.
```

QUESTION 4, 6.1, 7.1 SERVERMANAGER CLASS

```
unit uServerManager;

interface
  uses SysUtils, uTechnician, uServer;
//Question 4.1 - 1
 header correct
type TServerManager = class
  private
  //Question 4.2 - 4
   Both properties private
   Array of type servers with correct name
   Array size set to  50
    sArr : array[1..50] of TServer;
     size created correctly
    size : integer;
  public
    constructor Create;
    function allServers : string;
    function countServers(fault, role : string) : integer;
    procedure assignTechnicians();
    function findServer(loc , tid : string) : Boolean;
    function printMap(techID : string) : string;
end;

implementation

{ TServerManager }
//Question 4.3 - 9
 constructor header correct
constructor TServerManager.Create;
var
  infile : textfile;
  line, serverID, location, fault, role : string;
begin
  if FileExists('servers.txt') <> true then  handle exception
    begin
      WriteLn('File Missing');

    end
  else
    begin
       open file for reading
      AssignFile(infile, 'servers.txt');
      Reset(infile);

      size:=0;
       loop through all the lines

      while NOT EOF(inFile) do
        begin
          read the next line from the file
          ReadLN(infile, line);
          increment size
          Inc(size);
          split the line into the required parts
          serverID := Copy(line, 1, Pos('#', line) - 1);
          Delete(line, 1, Pos('#', line));

          location:= Copy(line, 1, Pos('#', line) - 1);
          Delete(line, 1, Pos('#', line));

          fault := Copy(line, 1, Pos('#', line) - 1);
          Delete(line, 1, Pos('#', line));

          role:= line;
          create Server object
          add to array
          sArr[size] := TServer.Create(serverID,location,fault,
role);
        end;
    end;
end;

//Question 4.4 - 5
 method header correct and return string
function TServerManager.allServers: string;
var
  i : integer ;
  output : string;
begin
     output := '';  string returned initialized
     loop through array
     for i := 1 to size do
      begin
        append to string with  extra blank line
        output := output + sArr[i].toString + #13#10 + #13#10;

      end;
      Result:=output;
end;

//Question 4.5 - 5
 header correct returns integer
function TServerManager.countServers(fault, role: string):
integer;
var
  count : integer;
  i : integer;
begin
  count:=0;
   loop through array
  for i := 1 to size do
    begin
     check if server fault
     role match parameters ignoring case
      if ( CompareText(fault, sArr[i].getFault) = 0 ) AND
```

```
       (CompareText(role, sArr[i].getRole) = 0 ) then
       begin
         count:= count + 1;      update count correctly
       end;
    end;
    Result:=count;
end;


//Question 6
//Question 6.1 - 11
 header correct
procedure TServerManager.assignTechnicians;
var
  infile : textfile;
  line, techID, name, roleSpeciality: string;
  experience : integer;
  tech : TTechnician;
  assigned: TArray<string>;
  j,k ,lim: integer;

begin
      if FileExists('servers.txt') <> true then
    begin
      WriteLn('File Missing');
    end
  else
    begin
      open file
      AssignFile(infile, 'technicians.txt');
      Reset(infile);
      loop through file
      while NOT EOF(inFile) do
        begin
          get lines and split
          ReadLN(infile, line);
          techID := Copy(line, 1, Pos('#', line) - 1);
          Delete(line, 1, Pos('#', line));

          name:= Copy(line, 1, Pos('#', line) - 1);
          Delete(line, 1, Pos('#', line));

          experience := StrToInt(Copy(line, 1, Pos('#', line) -
1));
          Delete(line, 1, Pos('#', line));

          roleSpeciality:= line;
          create Technician object
          tech := TTechnician.Create(techID,name,experience,
roleSpeciality);
          loop through all the servers

         checking that the limit of 4 servers per technician is not
exceeded
```

```
         using while looping structure
          lim:= 0;
          k:=1;

         while (lim < 4) and (k <= size) do
          begin
          check to see if the server role matches the technician role
and no technician has been assigned
            if
(CompareText(sArr[k].getRole(),tech.getRoleSpeciality()) = 0) and
(sArr[k].getAssignedTech() = nil) then
            begin
               assign technician to server using method in server class
              sArr[k].setAssignedTech(tech);
                increase number of server assigned
              lim := lim + 1;
            end;
            Inc(k);
          end;
        end;
    end;
end;
//Question 7

//Question 7.1 - 17
function TServerManager.findServer(loc , tid : string): Boolean;
var
  i : integer;
begin
  Result:=false;
   loop through servers
  for i := 1 to size do
    begin
          check if techid matches server techid
          if (sArr[i].getAssignedTech() <> nil) and
(sArr[i].getAssignedTech().getTechID() = tid) and (sArr[i].getLocation() = loc)
then
          begin
            Result:=true;
          end;
    end;
end;

function TServerManager.printMap(techID: string): string;
var
f : TextFile;
map ,loc: string;
i, col:integer;
row : char;

begin
   create a string for the map
  map:= '';
   create date stamp
   correct format
   added to map
  map:=  map + (FormatDateTime('YYYY/MM/dd HH:mm:ss', Now))  + #13#10; ;

    create and append column numbers
  for i := 1 to 15 do
    begin
      map:= map + #9 + IntToStr(i);
    end;
  map:= map + #13#10;
   create loop for row letters
  for row := 'A' to 'J' do
    begin
      map:= map + row;
        create loop for columns
      for col := 1 to 15 do
        begin
             check if a server location is found
            loc:= row + '' + IntToStr(col);
            if findServer(loc, techID) then
            begin
              map:= map + ' ' + #9 + 'X';  append x
            end
            else
            begin
              map:= map + ' ' + #9 + '*';  append *
            end;
        end;
      map:= map + #13#10;
    end;
     create file to write save map data with techID as file name
    AssignFile(f, techID + '.txt');
    ReWrite(f);
     write map data to file
    Writeln(f,map);
     close file
    CloseFile(f);
   return map
  Result:= map;end;

end.
```

QUESTION 5, 6.2 & 7.2 SERVERUI CLASS

```
//Question 5.1 - 1
 application class created
program ServerUI;
{$APPTYPE CONSOLE}
{$R *.res}
uses
  System.SysUtils,
  uTechnician in 'uTechnician.pas',
  uServer in 'uServer.pas',
  uServerManager in 'uServerManager.pas';
```

```
var
  sm : TServerManager;

begin
  try
    { TODO -oUser -cConsole Main : Insert code here }
   //Question 5.2 - 1
     create ServerManager object
    sm:= TServerManager.Create();  //Question 5.3 - 1
     allServer method called and displayed correctly
    WriteLn(sm.allServers());


    //Question 5.4 - 3
   //countServer method called using Constant value and display
correctly
   countServer method called
   using Constant value and called correctly
    WriteLn(sm.countServers('Temp' , TServer.ROLETYPE_CUSTOM)) ;

//Question 6.2 - 2
     assignedTechnicians called correctly and redisplay allServers
    sm.assignTechnicians();

    WriteLn(sm.allServers);
   //Question 7.2 - 2
   printMap called correctly
   correct techID used
    WriteLn(sm.printMap('T-D1'));

    ReadLn;
  except
    on E: Exception do
      Writeln(E.ClassName, ': ', E.Message);
  end;
end.
```