NATIONAL SENIOR CERTIFICATE EXAMINATION
NOVEMBER 2020

INFORMATION TECHNOLOGY: PAPER II

MARKING GUIDELINES

Time: 3 hours 120 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 SQL

QUESTION 1.1 [3]

```
SELECT *   FROM Client          -- 1
WHERE Age between 18 and 30     -- 1
ORDER BY Age                    -- 1
```

ALTERNATIVE:

```
SELECT *   FROM Client          -- 1
WHERE Age >= 18 AND Age <=30    -- 1
ORDER BY Age                    -- 1
```

QUESTION 1.2 [3]

```
SELECT * FROM Client            -- 1 for SELECT & FROM
WHERE PostCode LIKE             -- 1
'012%'                          -- 1     (Access: '012*')
```

ALTERNATIVE:
ACCESS & MYSQL

```
SELECT * FROM Client            -- 1 for SELECT & FROM
WHERE LEFT(PostCode, 3)         -- 1
= '012'                         -- 1
```

JAVADB

```
SELECT * FROM Client            -- 1 for SELECT & FROM
WHERE SUBSTR(PostCode,1, 3)     -- 1
= '012'                         -- 1
```

QUESTION 1.3 [5]

MYSQL:

```
SELECT ClientName, CONCAT(      -- 1 Join
  RIGHT(ClientName, 2)          -- 1 Right correct
  , FLOOR(11 +                  -- 1 Starting from 11 & FLOOR
    RAND()*                     -- 1 Random
    (17-11+1))                  -- 1 Correct range
) FROM Client
```

ACCESS:

```
SELECT ClientName,
  RIGHT(ClientName, 2)          -- 1 right correct
  &                             -- 1 Join
   Int(11+                      -- 1 Starting from 11 & INT
   (17-11+1)*                   -- 1 Correct range
   Rnd(ClientID))               -- 1 Random with different seed
FROM Client
```


JAVADB:

```
SELECT ClientName,
  SUBSTR(ClientName, LENGTH(ClientName)-1, 2)-- 1 right correct
  ||                                         -- 1 Join
  CHAR(INT(11 +                              -- 1 starting from 11 & INT
   (RANDOM()                                 -- 1 random
   *(17-11+1))))                             -- 1 Correct range
FROM Client
```

QUESTION 1.4 [3]

```
SELECT ClientID, Location FROM Appointment      -- 1
WHERE Location NOT IN                           -- 1
('Bergsig', 'Panorama', 'Highlands')            -- 1 Format
```

ALTERNATIVE:

```
SELECT ClientID, Location FROM Appointment    -- 1
WHERE Location <> 'Bergsig'    -- 1 <> used correctly (accept != JavaDB)
  AND Location <> 'Panorama'   -- 1 all 3 checked for correctly
  AND Location <> 'Highland'
```

QUESTION 1.5 [5]

```
SELECT counsellorName,
Rate
FROM Counsellor                       -- 1 for correct SELECT and FROM
WHERE Rate =                          -- 1
( SELECT MIN(Rate)                    -- 1
   FROM Counsellor                    -- 1
   WHERE Rate <> 0                    -- 1 (accept != JavaDB)
)
```

QUESTION 1.6 [6]

```
SELECT Location, COUNT(*)              -- 1
FROM Appointment
WHERE AppointmentDate > CURRENT_DATE   -- (ACCESS: NOW())
                                       -- 1 current date (accept >=)
GROUP BY Location                      -- 2
HAVING COUNT(*) > 15                   -- 2
```


QUESTION 1.7 [5]

```
SELECT AppointmentID, Appointment.ClientID, ClientName, CounsellorName,
  Rate * 0.75                                        -- 1
FROM Client, Counsellor, Appointment                 -- 1
WHERE Client.ClientID = Appointment.ClientID         -- 1
  AND Counsellor.CounsellorID = Appointment.CounsellorID   -- 1
  AND age < 15     -- 1
```

QUESTION 1.8 [4]

```
UPDATE Appointment                      -- 1
SET Location = 'Middelburg'             -- 1
WHERE Location = 'Bergsig'              -- 1
  AND AppointmentDate >= '2020-10-24'   -- 1 correct date
ACCESS: AND AppointmentDate >= #10/24/2020#
```

QUESTION 1.9 [6]

```
INSERT INTO Appointment  -- 1
(ClientID, CounsellorID, AppointmentDate, Location)
                         -- 1 list includes all fields asked
SELECT 27, CounsellorID, '2020-10-20', Location
                         -- 1 order matches
                         -- 1 fields counsellorID, Location included
                         -- 1 format correct for others
                         -- ACCESS #10/20/2020# for date
FROM Appointment
WHERE AppointmentID = 18 -- 1
```


SECTION B OBJECT-ORIENTED PROGRAMMING

JAVA

QUESTION 2 Client.java

```
// Question 2.1 - 4
// class created
public class Client {
    private String clientName;            // private
    private String preferredCounsellor;   // correct types
    private int earliestHour;             // correct names

    // Question 2.2 - 4
    // correct header and names/type/order as asked
    public Client(String inCN, String inPCS, int inEH) {
        clientName = inCN;            // string fields set to parameters
        preferredCounsellor = inPCS;

        if (inEH > 16)        // earliestHour set to 16 if inEH > 16
        {
            earliestHour = 16;
        }
        else                  // set to inEH otherwise
        {
            earliestHour = inEH;
        }
    }

    // Question 2.3 - 2
    // correct headers for all three methods
    // correct returns for all three methods
    public String getClientName() {
        return clientName;
    }

    public String getPreferredCounsellor() {
        return preferredCounsellor;
    }

    public int getEarliestHour() {
        return earliestHour;
    }

    @Override
    // Question 2.4 - 4
    // correct header
    public String toString() {
        return clientName + "\t" + preferredCounsellor + "\tEarliest "
               + earliestHour + ":00";
        // correct return
        // all elements included
        // format correct including tabs and :00
    }
}
```


QUESTION 3

TimeSlot.java

```
// Question 3.1 - 3
// class created correctly
public class TimeSlot {
    private String counsellor;       // private
    private int startHour;           // types/names correct and as given
    private boolean isAvailable = true;

    // Question 3.2 – 4
    // header correct
    // only takes in string & int – not boolean
    // boolean default to true (in constructor or field declaration)
    public TimeSlot(String inCS, int inSH) {
        counsellor = inCS;
        startHour = inSH;    // set other fields
    }

    // Question 3.3 - 2
    // set headers correct
    // set assigned correctly
    public void setIsAvailable(boolean isAvailable) {
        this.isAvailable = isAvailable;
    }

    // Question 3.4 - 1
    // all three get methods correct
    public String getCounsellor() {
        return counsellor;
    }

    public int getStartHour() {
        return startHour;
    }

    public boolean getIsAvailable() {
        return isAvailable;
    }


    // Question 3.5 - 2
    // correct header with private
    private int getEndHour() {
        // return startHour + 1
        return startHour + 1;
    }
```



```
    // Question 3.6 - 4
    @Override
    public String toString() {            // correct header
         // use getEndHour to get end time
         // include other fields
         // format correct
        return counsellor + ": " + startHour + ":00 - " + getEndHour()
               + ":00";
    }
}
```

QUESTION 4, 5, 6 AND 7

SlotManager.java

```
import java.io.*;

// Question 4.1 - 4
public class SlotManager {                    // class created correctly
  private Client[] cArr = new Client[20];     // client array created
  private TimeSlot[] tArr = new TimeSlot[40]; // timeslot array created
                                              // both private with
                                              // correct sizes

  // Question 4.2 - 8
  public SlotManager() {
   try {
    // read file correctly
    BufferedReader br = new BufferedReader(new FileReader("clients.txt"));

    // correct loop structure
    // correct value to enable correct read of 20 clients in 60 lines
    for (int i = 0; i < cArr.length; i++) {
        String clientname = br.readLine();      // read three lines
        String preferredcs = br.readLine();
        int earliest = Integer.parseInt(br.readLine());
                                                // correct conversion
        // create client
        Client c = new Client(clientname, preferredcs, earliest);
        // add new client to the correct slot
        cArr[i] = c;
        // correct update (can be part of for header)
     }
    } catch (FileNotFoundException ex)
    {
       System.out.println("File not found");
    } catch (IOException ex)
    {
       System.out.println("IO error");
    }
  }
```



```
// Question 4.3 - 4
// correct header
public String displayAllClients() {
  String temp = "";
  // loop to go through correct array and correct length
  for (int i = 0; i < cArr.length; i++) {
      // build up a string correctly
      temp += cArr[i] + "\n";
  }
  // return the built-up string
  return temp;
}

// Question 6.1 - 11
// correct header
public void generateTimeSlots() {
 try {
  // open file to read correctly
  BufferedReader br = new BufferedReader(new
                        FileReader("counsellors.txt"));

  String line = br.readLine();     // read one line

  // split counsellor names correctly
  String[] csArr = line.split(",");

  int counter = 0;      // create and initialise counter correctly

  // use loops to generate timeslots successfully
  // timeslots loaded
  for (int i = 0; i < csArr.length; i++) {
    for (int j = 8; j <= 16; j++) {
      // exclude lunch hour
      if (j!= 12) {
        // time slot created with correct parameters
        TimeSlot t = new TimeSlot(csArr[i], j);
        // added to correct array at correct position
        tArr[counter] = t;
        counter++;               // increment counter after
      }
    }
  }
 } catch (FileNotFoundException ex)
 {
     System.out.println("File not found");
 } catch (IOException ex)
 {
     System.out.println("IO error");
 }
}
```



```
  // Question 6.2 - 4
  // correct header
  public String displayAllAvailableTimeSlots() {
    String temp = "";
    // loop to go through correct array length
    for (int i = 0; i < tArr.length; i++) {
      // check if slot is available
      if (tArr[i].getIsAvailable()) {
        // combine each time slot in correct format
        temp += tArr[i] + "\n";
      }
    }

    return temp;
  }

  // Question 7.1 - 13
  // header correct
  public String generateBookedSlots() {
    // create header with new line
    String temp = "Appointments: \n";

    // loop to go through client array
    for (int c = 0; c < cArr.length; c++) {
      // loop to go through timeslot array
      for (int t = 0; t < tArr.length; t++) {
        // check timeslot counsellor against client's preferred one
        if (tArr[t].getCounsellor().equals(
              cArr[c].getPreferredCounsellor())) {
          // check start hour against client
          if (tArr[t].getStartHour() >= cArr[c].getEarliestHour())  {
            // check if timeslot is available
            // correct conditional link (also accept and)
            if (tArr[t].getIsAvailable()) {
              // set matched timeslot's available to false
              tArr[t].setIsAvailable(false);

              // include just the information requested (and no more)
              // formatting correct
              temp = temp + cArr[c].getClientName() + " ("
                    + cArr[c].getEarliestHour() + ") to see "
                    + tArr[t] + "\n";
              break;  // break when found (or use flag)
            }
          }
        }
      }
    }

    return temp;  // return correct string

  }
}
```

ALTERNATIVE WITH WHILE AND FLAG

```
// Question 7.1 - 13
// header correct
public String generateBookedSlots() {
  // create header with new line
  String temp = "Appointments: \n";

  // loop to go through client array
  for (int c = 0; c < cArr.length; c++) {

    int t = 0;
    boolean found = false;
    // loop to go through timeslot array (check increment)
    while (t < tArr.length && found == false) {
      // check timeslot counsellor against client's preferred one
      if (tArr[t].getCounsellor().equals(
            cArr[c].getPreferredCounsellor())) {
        // check start hour against client
        if (tArr[t].getStartHour() >= cArr[c].getEarliestHour())  {
          // check if timeslot is available
          // correct conditional link (also accept and)
          if (tArr[t].getIsAvailable()) {
            // set matched timeslot's available to false
            tArr[t].setIsAvailable(false);

            // include just the information requested (and no more)
            // formatting correct
            temp = temp + cArr[c].getClientName() + " ("
                  + cArr[c].getEarliestHour() + ") to see "
                  + tArr[t] + "\n";
            found = true;  // use flag
          }
        }
      }
      t++;
    }
  }

  return temp;  // return correct string

}
```

CounsellingUI.java

```
import java.io.IOException;

public class CounsellingUI {
    // Question 5.1 - 1
    // text-based interface for input/output

    public static void main(String[] args) throws IOException {
        // Question 5.2 - 1
        // SlotManager created
        SlotManager sm = new SlotManager();

        // Question 5.3 - 1
        // display Clients Array correctly
        System.out.println(sm. displayAllClients());

        // Question 6.3 - 2
        // call generateTimeSlots() correctly
        sm.generateTimeSlots();
        // call to displayAllAvailableTimeSlots
        System.out.println(sm.printAllAvailableTimeSlots());

        // Question 7.2 - 1
        // method called and returned string printed
        System.out.println(sm.generateBookedSlots());
    }

}
```

DELPHI

QUESTION 2

uClient.pas

```
unit uClient;

interface
  uses SysUtils;

// Question 2.1 - 4
// class created
type TClient = class
    private                          // private
      clientName : string;           // correct types
      preferredCounsellor : string; // correct names
      earliestHour : integer;
    public
      constructor Create(inCN: string; inPCS: string; inEH: integer);
      function getClientName() : string  ;
      function getPreferredCounsellor(): string;
      function getEarliestHour() : integer;
      function toString() : string;
  end;

implementation
```



```
  // Question 2.2 – 4
  // correct header and names/type/order as asked
  constructor TClient.Create(inCN: string; inPCS: string; inEH: integer);
  begin
    clientName := inCN;             // string fields set to parameters
    preferredCounsellor := inPCS;

    if (inEH > 16) then    // earliestHour set to 16 if in EH > 16
      begin
        earliestHour := 16;
      end
    else                   // set to inEH otherwise
    begin
      earliestHour := inEH;
    end;
  end;

  // Question 2.3 - 2
  // correct headers for all three methods
  // correct returns for all three methods
  function TClient.getClientName() : string;
  begin
    Result := clientName;
  end;

 function TClient.getPreferredCounsellor() : string;
  begin
    Result := preferredCounsellor;
  end;

 function TClient.getEarliestHour() : integer;
  begin
    Result := earliestHour;
  end;

  // Question 2.4 - 4
  function TClient.toString() : string;     // correct header
  begin
    Result := clientName + #9 + preferredCounsellor + #9 + 'Earliest '
              + IntToStr(earliestHour) + ':00';
         // correct return
         // all elements included
         // format correct including tabs and :00
  end;
end.
```


QUESTION 3

uTimeSlot.pas

```
unit uTimeSlot;

interface
  uses SysUtils;

// Question 3.1 - 3
// class created correctly
type TTimeSlot = class
    private                   // private
      counsellor : string;    // types/names correct and as given
      startHour : integer;
      isAvailable : boolean;
      function getEndHour(): integer;
    public
      constructor Create(inCS: string; inSH: integer);
      procedure setIsAvailable(inIA: boolean);
      function getCounsellor() : string  ;
      function getStartHour(): integer;
      function getIsAvailable(): boolean;
      function toString() : string;
  end;

implementation
  // Question 3.2 – 4
  // header correct
  // only takes in string and int, not Boolean
    // boolean default to true (in constructor or field declaration)
  constructor TTimeslot.Create(inCS: string; inSH: integer);
  begin
    counsellor := inCS;     // set other fields
    startHour := inSH;
    isAvailable := true;
  end;


  // Question 3.3 - 2
    // set headers correct
    // set assigned correctly

  procedure TTimeSlot.setIsAvailable(inIA: boolean);
  begin
    isAvailable := inIA;
  end;
```



```
  // Question 3.4 - 1
    // all three  get methods correct
  function TTimeSlot.getCounsellor() : string;
  begin
    Result := counsellor;
  end;

  function TTimeSlot.getStartHour() : integer;
  begin
    Result := startHour;
  end;

  function TTimeSlot.getIsAvailable() : boolean;
  begin
    Result := isAvailable;
  end;


  // Question 3.5 - 2
  // correct header with private (check above)
  function TTimeSlot.getEndHour() : integer;
  begin       // return startHour + 1
    Result := startHour + 1;
  end;


  // Question 3.6 - 4
  function TTimeSlot.toString() : string;   // correct header
  begin
    Result := counsellor + ': ' + IntToStr(startHour) + ':00 - '
              + IntToStr(getEndHour()) + ':00';
         // use getEndHour to get end time
         // include other fields
         // format correct
  end;
end.
```

QUESTION 4, 5, 6 AND 7

uSlotManager.pas

```
unit uSlotManager;

interface
uses
    SysUtils, uClient, uTimeSlot;

  // Question 4.1 - 4
  type TSlotManager = class              // class created correct
    private
      cArr : array[1..20] of TClient;    // client array created
      tArr : array[1..40] of TTimeSlot;  // timeslot array created
                                      // both private with correct sizes
    public
      constructor Create();
      function displayAllClients() : string;
```

```
      procedure generateTimeSlots();
      function displayAllAvailableTimeSlots() : string;
      function generateBookedSlots() : string;
  end;
implementation

  // Question 4.2 - 8
  constructor TSlotManager.Create();
  var
    inFile : TextFile;
    clientname, preferredcs, earliestst : string;
    earliest, i : integer;
  begin
    AssignFile(inFile, 'clients.txt');  // read file correctly
    Reset(inFile);

    // correct loop structure
    // correct value to enable correct read of 20 clients in 60 lines
    for i :=1 to length(cArr) do
    begin
      Readln(inFile, clientname);        // read three lines
      Readln(inFile, preferredcs);
      Readln(inFile, earliestst);
      earliest := StrToInt(earliestst);  // correct conversion

      // create client
      // add new client to the correct slot
      cArr[i] := TClient.Create(clientname, preferredcs, earliest);
      // correct update (can be part of header)
    end
  end;


  // Question 4.3 - 4
  // correct header
  function TSlotManager.displayAllClients() : string;
  var
    i : integer;
  begin
    Result := '';
    // loop to go through correct array and correct length
    for i:=1 to length(cArr) do
      // build up a string correctly
      Result := Result + cArr[i].toString() + #10#13;
      // return the built-up string
  end;


  // Question 6.1 - 11
  // correct header
  procedure TSlotManager.generateTimeSlots();
  var
    inFile : TextFile;
    line, counsellor : string;
```

```
    i, h, tcount : integer;
  begin
    //  open file to read correctly
    AssignFile(inFile, 'counsellors.txt');
    Reset(inFile);

    tcount := 1;              // create and initialise counter correctly
    Readln(inFile, line);     // read one line

    // use loops to generate timeslots successfully
    // timeslots loaded
    for i:=1 to 5 do
    begin
      if (i <5) then
      begin
        // split counsellor names correctly
        counsellor := Copy(line, 1, Pos(',', line) -1 );
        Delete(line, 1, Pos(',', line));
      end
      else
        counsellor := line;

      for h:=8 to 16 do
      begin
        // exclude lunch hour
        if (h <> 12) then
        begin
          // time slot created with correct parameters
          // added to correct array at correct position
          tArr[tcount] := TTimeSlot.Create(counsellor, h);
          tcount := tcount + 1;   // increment counter after
        end;
      end;
    end;
  end;


  // Question 6.2 - 4
  // correct header
  function TSlotManager.displayAllAvailableTimeSlots() : string;
  var
    i : integer;
  begin
    Result := '';
    // loop to go through correct array length
    for i:=1 to length(tArr) do
      // check if slot is available
      if (tArr[i].getIsAvailable()) then
        // include each time slot in correct format
        Result := Result + tArr[i].toString();
  end;
```

```
  // Question 7.1 - 13
  // header correct
  function TSlotManager.generateBookedSlots() : string;
  var
    c, t : integer;
  begin
    // create header with new line
    Result := 'Appointments: ' + #10#13;

    // loop to go through client array
    for c:=1 to length(cArr) do
      // loop to go through timeslot array
      for t:=1 to length(tArr) do
        // check timeslot counsellor against client's preferred one
        if (tArr[t].getCounsellor()
              = cArr[c].getPreferredCounsellor()) then
          // check start hour against client
          if (tArr[t].getStartHour() >= cArr[c].getEarliestHour()) then
            // check if timeslot is available
            // correct conditional link (also accept and)
            if (tArr[t].getIsAvailable()) then
            begin
              // set matched timeslot's available to false
              tArr[t].setIsAvailable(false);

              // include just the information requested (and no more)
              // formatting correct
              Result := Result + cArr[c].getClientName() + ' ('
                        + IntToStr(cArr[c].getEarliestHour()) + ') to see '
                        + tArr[t].toString() + #10#13;
              break;    // break when found (or use flag)
            end

            // return correct string
  end;
end.
```

ALTERNATIVE WITH WHILE AND FLAG

```
  // Question 7.1 - 13
  // header correct
  function TSlotManager.generateBookedSlots() : string;
  var
    c, t : integer;
    found : boolean;
  begin
    // create header with new line
    Result := 'Appointments: ' + #10#13;

    // loop to go through client array
    for c:=1 to length(cArr) do
    begin
      // loop to go through timeslot array
      t := 0;
      found := false;
```

```
      while (t < length(tArr)) and (found = false) do
      begin
        // check timeslot counsellor against client's preferred one
        if (tArr[t].getCounsellor()
              = cArr[c].getPreferredCounsellor()) then
          // check start hour against client
          if (tArr[t].getStartHour() >= cArr[c].getEarliestHour()) then
            // check if timeslot is available
            // correct conditional link (also accept and)
            if (tArr[t].getIsAvailable()) then
            begin
              // set matched timeslot's available to false
              tArr[t].setIsAvailable(false);

              // include just the information requested (and no more)
              // formatting correct
              Result := Result + cArr[c].getClientName() + ' ('
                        + IntToStr(cArr[c].getEarliestHour()) + ') to see '
                        + tArr[t].toString() + #10#13;
              found := true;    // use flag
            end;
            t:=t+1;
      end;
    end;         // return correct string
  end;
```

CounsellingUI

```
program HopeProject;

{$APPTYPE CONSOLE}

{$R *.res}

uses
  System.SysUtils,
  uClient in 'uClient.pas',
  uTimeSlot in 'uTimeSlot.pas',
  uSlotManager in 'uSlotManager.pas';

var
  input : string;
  sm : TSlotManager;
begin
  // Question 5.1 - 1
  // text-based interface to input/output
  try
    // Question 5.2 - 1
    // SlotManager created
    sm := TSlotManager.Create();

    // Question 5.3 - 1
    // call displayAllClients() correctly
    WriteLn(sm.displayAllClients());
```



```
    // Question 6.3 - 2
    // call generateTimeSlots() correctly
    sm.generateTimeSlots();
    // call displayAllAvailableTimeSlots()
    WriteLn(sm. displayAllAvailableTimeSlots());


    // Question 7.2 - 1
    // method called and returned string printed
    WriteLn(sm.generateBookedSlots());


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

OUTPUT

SECTION A

QUESTION 1.1

ClientID	ClientName	Age	PostCode
28	Cynthia Fourie	18	1500
8	Lindewe Khoza	18	1500
16	Maria Nkosi	21	0152
3	Steve Jacobs	21	0125
33	Bianca Abrahams	22	0160
26	Zandile Methembu	22	1600
2	Leo Sithole	22	1240
31	Al Naidoo	23	0127
23	Willem du Plessis	23	1600
4	Pat Khumalo	23	1251
30	Bongiwe Mokoena	25	0160
7	Sibongile Ngcobo	25	1240
37	Hendrick van Zyl	27	0180
15	Peter Zwane	30	1240

QUESTION 1.2

ClientID	ClientName	Age	PostCode
3	Steve Jacobs	21	0125
6	Musi Mahlangu	38	0120
18	Patricia Williams	17	0120
31	Al Naidoo	23	0127
35	Moses Sibisi	31	0121
38	Lucky Shabangu	16	0123

QUESTION 1.3

Note that the last two digits are randomly generated

ClientName	Expr1001
John Dlamini	ni16
Leo Sithole	le15
Steve Jacobs	bs16
Pat Khumalo	lo11
Sipho Nkosi	si12
Musi Mahlangu	gu11
Sibongile Ngcobo	bo11
Lindewe Khoza	za13
Johannes Botha	ha11
Bongani Radebe	be11
Caleb Pillay	ay14
Anna Mthembu	bu15
Thabo Baloyi	yi14
Mpho Sithole	le16
Peter Zwane	ne11
Maria Nkosi	si12
Samuel Naidoo	oo15
Patricia Williams	ms14
Joseph Mkhize	ze13
Zenele Ndlovu	vu12
Daniel Dlamini	ni15
Jabu Chauke	ke17
Willem du Plessis	is14
Jan Venter	er11
Linda Gumede	de16
Zandile Methembu	bu13
Robert Maseko	ko14
Cynthia Fourie	ie14
Nomsa Baloyi	yi12
Bongiwe Mokoena	na13
Al Naidoo	oo11
Phindile Ntuli	li15
Bianca Abrahams	ms12
Freddy Chetty	ty17
Moses Sibisi	si11
Vusi Khumalo	lo14
Hendrick van Zyl	yl12
Lucky Shabangu	gu17

QUESTION 1.4

ClientID	Location
2	Greenside
5	Greenside
24	Greenside
29	Greenside
36	Greenside
4	Greenside
15	Greenside
27	Greenside
2	Greenside
24	Greenside
29	Greenside
14	Greenside
4	Greenside
2	Greenside
24	Greenside
29	Greenside
4	Greenside
15	Greenside
27	Greenside
14	Greenside
29	Greenside
4	Greenside
15	Greenside

QUESTION 1.5

CounsellorName	Rate
Siyanda Mabuza	120
Joshua Hendricks	120
Linda September	120

QUESTION 1.6

Correct output when run on 16 October 2020.
A different current date will change the output.

Location	NumberAppointments
Bergsig	22
Panorama	17

QUESTION 1.7

AppointmentID	ClientID	ClientName	CounsellorName	Expr1004
6	5	Sipho Nkosi	Joshua Hendricks	90
35	14	Mpho Sithole	Thabo Matlala	112.5
69	14	Mpho Sithole	Thabo Matlala	112.5
24	25	Linda Gumede	Matthew Kunene	135
44	25	Linda Gumede	Matthew Kunene	135
8	29	Nomsa Baloyi	Joshua Hendricks	90
34	29	Nomsa Baloyi	Joshua Hendricks	90
57	29	Nomsa Baloyi	Joshua Hendricks	90
70	29	Nomsa Baloyi	Thabo Matlala	112.5
9	36	Vusi Khumalo	Joshua Hendricks	90

SECTION B

FINAL OUTPUT

```
John Dlamini        Vernon Booysen      Earliest 10:00
Leo Sithole         Matthew Kunene      Earliest 13:00
Steve Jacobs        Vernon Booysen      Earliest 8:00
Pat Khumalo         Heather Modise      Earliest 6:00
Sipho Nkosi         Vernon Booysen      Earliest 16:00
Musi Mahlangu       Joshua Hendricks    Earliest 9:00
Sibongile Ngcobo    Joshua Hendricks    Earliest 8:00
Lindewe Khoza       Siyanda Mabuza      Earliest 10:00
Johannes Botha      Vernon Booysen      Earliest 10:00
Bongani Radebe      Heather Modise      Earliest 9:00
Caleb Pillay        Vernon Booysen      Earliest 9:00
Anna Mthembu        Siyanda Mabuza      Earliest 11:00
Thabo Baloyi        Heather Modise      Earliest 12:00
Mpho Sithole        Matthew Kunene      Earliest 12:00
Peter Zwane         Vernon Booysen      Earliest 9:00
Maria Nkosi         Joshua Hendricks    Earliest 14:00
Samuel Naidoo       Vernon Booysen      Earliest 9:00
Patricia Williams   Heather Modise      Earliest 8:00
Joseph Mkhize       Matthew Kunene      Earliest 7:00
Zenele Ndlovu       Siyanda Mabuza      Earliest 14:00

Vernon Booysen: 8:00 - 9:00
Vernon Booysen: 9:00 - 10:00
Vernon Booysen: 10:00 - 11:00
Vernon Booysen: 11:00 - 12:00
Vernon Booysen: 13:00 - 14:00
Vernon Booysen: 14:00 - 15:00
Vernon Booysen: 15:00 - 16:00
Vernon Booysen: 16:00 - 17:00
Matthew Kunene: 8:00 - 9:00
Matthew Kunene: 9:00 - 10:00
Matthew Kunene: 10:00 - 11:00
Matthew Kunene: 11:00 - 12:00
Matthew Kunene: 13:00 - 14:00
Matthew Kunene: 14:00 - 15:00
Matthew Kunene: 15:00 - 16:00
```



```
Matthew Kunene: 16:00 - 17:00
Heather Modise: 8:00 - 9:00
Heather Modise: 9:00 - 10:00
Heather Modise: 10:00 - 11:00
Heather Modise: 11:00 - 12:00
Heather Modise: 13:00 - 14:00
Heather Modise: 14:00 - 15:00
Heather Modise: 15:00 - 16:00
Heather Modise: 16:00 - 17:00
Siyanda Mabuza: 8:00 - 9:00
Siyanda Mabuza: 9:00 - 10:00
Siyanda Mabuza: 10:00 - 11:00
Siyanda Mabuza: 11:00 - 12:00
Siyanda Mabuza: 13:00 - 14:00
Siyanda Mabuza: 14:00 - 15:00
Siyanda Mabuza: 15:00 - 16:00
Siyanda Mabuza: 16:00 - 17:00
Joshua Hendricks: 8:00 - 9:00
Joshua Hendricks: 9:00 - 10:00
Joshua Hendricks: 10:00 - 11:00
Joshua Hendricks: 11:00 - 12:00
Joshua Hendricks: 13:00 - 14:00
Joshua Hendricks: 14:00 - 15:00
Joshua Hendricks: 15:00 - 16:00
Joshua Hendricks: 16:00 - 17:00


// This list may have different times depending on the algorithm
// used by the candidate. However, they should be in line with the
// earliest time for each client (see above).
Appointments:
John Dlamini (10) to see Vernon Booysen: 10:00 - 11:00
Leo Sithole (13) to see Matthew Kunene: 13:00 - 14:00
Steve Jacobs (8) to see Vernon Booysen: 8:00 - 9:00
Pat Khumalo (6) to see Heather Modise: 8:00 - 9:00
Sipho Nkosi (16) to see Vernon Booysen: 16:00 - 17:00
Musi Mahlangu (9) to see Joshua Hendricks: 9:00 - 10:00
Sibongile Ngcobo (8) to see Joshua Hendricks: 8:00 - 9:00
Lindewe Khoza (10) to see Siyanda Mabuza: 10:00 - 11:00
Johannes Botha (10) to see Vernon Booysen: 11:00 - 12:00
Bongani Radebe (9) to see Heather Modise: 9:00 - 10:00
Caleb Pillay (9) to see Vernon Booysen: 9:00 - 10:00
Anna Mthembu (11) to see Siyanda Mabuza: 11:00 - 12:00
Thabo Baloyi (12) to see Heather Modise: 13:00 - 14:00
Mpho Sithole (12) to see Matthew Kunene: 14:00 - 15:00
Peter Zwane (9) to see Vernon Booysen: 13:00 - 14:00
Maria Nkosi (14) to see Joshua Hendricks: 14:00 - 15:00
Samuel Naidoo (9) to see Vernon Booysen: 14:00 - 15:00
Patricia Williams (8) to see Heather Modise: 10:00 - 11:00
Joseph Mkhize (7) to see Matthew Kunene: 8:00 - 9:00
Zenele Ndlovu (14) to see Siyanda Mabuza: 14:00 - 15:00
```