NATIONAL SENIOR CERTIFICATE EXAMINATION
NOVEMBER 2018

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

QUESTION 1

Question 1.1 (4)

```
SELECT *
FROM Tourist
WHERE Email LIKE '%seattletimes.com'     -- JavaDB and MySQL

WHERE Email LIKE '*seattletimes.com'     -- Access
```

ALTERNATIVE:
JavaDB: `WHERE SUBSTR(Email, LENGTH(Email)-16+1, 16 ) = 'seattletimes.com'`

Access: `WHERE RIGHT(Email, 16) = 'seattletimes.com'`

Question 1.2 (3)

```
UPDATE Tourist
SET Hotel = 'Three Seasons Hotel'
WHERE Hotel = 'Lunar Hotel'
```

Question 1.3 (6)

```
SELECT ExcursionName,
EndHour - StartHour
AS Duration
FROM Excursion
WHERE EndHour <= 11          -- alternative < 12
AND EndHour - StartHour <= 3
```

Question 1.4 (6)

```
SELECT Hotel ,
COUNT(*)
FROM Tourist
GROUP BY  Hotel
HAVING COUNT(*)>=3
```


Question 1.5 (6)

```
SELECT TouristName
FROM Tourist
WHERE TouristID NOT IN
( SELECT TouristID
  FROM Booking)
ORDER BY TouristName
```

ALTERNATIVE:

```
SELECT TouristName
FROM Tourist LEFT JOIN Booking
ON Tourist.TouristID = Booking.TouristID
WHERE ExcursionID IS NULL
ORDER BY TouristName
```

Question 1.6 (6)

JavaDB:

```
SELECT ExcursionName,
     SUBSTR(ExcursionName, 1, 3)
     ||
     CHAR (INT(RANDOM()*90)+10 )   correct formula
FROM Excursion
```

MySQL

```
SELECT ExcursionName,
      CONCAT(
            SUBSTR( ExcursionName, 1, 3 ) ,
            FLOOR( RAND( ) *90 ) +10 )  correct formula
FROM Excursion
```

Access

```
SELECT ExcursionName,
      LEFT(ExcursionName, 3)
      &
      INT (Rnd(ExcursionID) *90 + 10)  correct formula
                    -- Access Rnd must have different seed
FROM Excursion
```

Question 1.7 (9)

```
INSERT INTO Booking (TouristID, ExcursionID, CostCharged, ExcursionDate)
                                  -- 2 correct table and all fields present
SELECT TouristID, ExcursionID,    -- 1 correct order (matches insert)
CurrentCost+Surcharge,
CURRENT_DATE        -- 1  current date or NOW() for Access (not hard code)
FROM Excursion, Tourist                -- subtract 1 if join is present
WHERE ExcursionID IN ( 1, 7, 13, 5 )   -- or 4 conditions with OR
AND Hotel = 'President Hotel'
```


JAVA SOLUTION

QUESTION 2: STOP CLASS

```
// Q2.1 - (1)     ✓
public class Stop {

    // Q2.2 - (3)
    private String stopName;
    private String routeCodes;
    private int stopType;

    // Q2.3 - (3)
    public static final int STOPTYPE_CAFE = 1;
    public static final int STOPTYPE_SHELTER = 2;
    public static final int STOPTYPE_EXPRESS = 3;
    public static final int STOPTYPE_OTHER = 4;

    // Q2.4 – (7)     ✓
    public Stop(String inStopName, String inRouteCodes, int inStopType)
    {
        stopName = inStopName;
        routeCodes = inRouteCodes;

        if (inStopType == STOPTYPE_CAFE ||
            inStopType == STOPTYPE_SHELTER ||
            inStopType == STOPTYPE_EXPRESS)
        {
            stopType = inStopType;
            // also accept if each one set individually
        }
        else
        {
            stopType = STOPTYPE_OTHER;
        }
    }

    // Q2.5 – (3)
    public String getStopTypeName() ✓
    {
        switch (stopType) ✓      // if/else if/else acceptable
        {
            case STOPTYPE_CAFE:
                return "cafe";
            case STOPTYPE_EXPRESS:
                return "express";
            case STOPTYPE_SHELTER:
                return "shelter";
            default:
                return "other";
        }
    }
```

✓ Private
✓ Correct Type
✓ Named as asked

✓ Final/const
✓ Named correctly
✓ Int values correct

✓ Properties set correctly

✓ Comparison correct
✓ Check against constants
✓ Use OR correctly

✓ Set only if valid

✓ otherwise set to TYPE other (not literal int)

✓ return correct string based on constants

```
// Q2.6 – (3)                                    ✓
    public boolean isPartOfRoute(char r)
    {          ✓A1                                ✓A1
        return (routeCodes.contains("" + r));

        // alternatives
        // use if statement to work out what to return
        if (routesCode.contains("" + r))
        {                                  ✓A2
            return true;
        }
        else                                                    ✓A2
        {
            return false;
        }


        // use indexOf      ✓A3                          ✓A3
        return (routeCodes.indexOf(r) > 0);

        // use indexOf with if          ✓A4
        if (routesCodes.indexOf(r) > 0)
        {
           return true;
        }                                                       ✓A4
        else
        {
           return false;
        }
    }


    // Q2.7 – (3)          ✓
    public String toString()
    {                                                   ✓all content included
        return getStopTypeName() + "\t" + stopName;     ✓correct format
    }

}
```

QUESTIONS 3 AND 5: ROUTE CLASS

```
// Q3.1 – (3)   ✓
public class Route {
    private char routeCode;
    private boolean isCircular;
    private Stop[] stops;

    // Q3.2 – (2)
    public Route(char inRouteCode, boolean inIsCircular)
    {
        routeCode = inRouteCode;
        isCircular = inIsCircular;
    }

    // Q3.3 – (2)
    public void setStops(Stop[] inStops)
    {
        stops = inStops;
    }

    // Q3.4 – (1)
    public char getRouteCode()
    {
        return routeCode;
    }

    // Q3.5 – (5)
    public Stop getStopAt(int num)
    {
        if (num >= 0 && num < stops.length)
        {
            return stops[num];
        }
        else return null

         // also accept if argument starts from 1 as long as slot
         //processing is correct and Q7.1 also matches
        /*
        if (num >=1  && num <= stops.length)
        {
            return stops[num-1];
        }
        else return null;
        */

    }
```

✓ Private
✓ Correct Type
✓ header
✓ set correctly
✓ header
✓ set correctly
✓
✓
✓ A1
✓ A1
✓ A1 - return object
✓ A1 return null for not valid
✓ A2
✓ A2
✓ A2 - return object
✓ A2 return null for not valid

```
    // Q5 - (9)
    public String toString()
    {
      String toRet = routeCode + " - ";

      if (stops.length == 0)          // also accept <= 0
      {
          toRet += "Invalid";
      }
      else
      {
           if (isCircular)
           {
              toRet += "Circular";
           }
           else
           {
              toRet += "Linear";
           }

           for (int i = 0; i < stops.length; i++)
           {
              toRet += "\n-> " + (i+1) + "\t" + stops[i];
           }

           if (isCircular)
           {
              toRet += "\n-> 1\t" + stops[0];
           }
           else
           {
              for (int i = stops.length -2 ; i >= 0; i--)
              {
                toRet += "\n-> " + (i+1) + "\t" +  stops[i];
              }
           }
      }
      return toRet;
    }

}
```

✓ ✓ **Efficiency mark:**
2 – no route processing for invalid stop
1 – invalid check is done, but there is unnecessary processing/checking of route type
0 – no invalid check done

✓ add circular / linear correctly

✓ include stops

✓ **Efficiency mark** – code to print out the first part of the array (where circular / linear arrays have common format) appears only once in the code i.e. no part of it appears in multiple locations.

✓ include first stop if circular

✓ loop to include stop in reverse from second last slot

✓ all stops returned in correct format

Questions 4 and 7.1 : TourManager Class

```
import java.io.*;

// Q4.1 – (1)          ✓
public class TourManager {

    // Q4.2 – (3)
    private Stop[] allStops = new Stop[100];
    private int stopCount = 0;

    // Q4.3 – (9)
    public StopManager(String fn)
    {
        try
        {
            BufferedReader br = new BufferedReader(new FileReader(fn));


            String line = br.readLine();

            while (line != null)
            {
                String[] tokens = line.split(",");
                String sName = tokens[0];
                int sType = Integer.parseInt(tokens[1]);
                String rCodes = tokens[2];

                allStops[stopCount] = new Stop(sName, rCodes, sType);
                // 1 L1 increment counter
                stopCount++;

                line = br.readLine();

            }
        } catch (Exception ex) { // or throws exception
            System.out.println("File not found");
        }
    }


    // Q4.4 – (10)
    public Route getRouteWithCode(char inCode, boolean inIsCircular)
    {
        Route r = new Route(inCode, inIsCircular);
```

✓ Private
✓ Correct Type
✓ Correct initial values

✓ correct method header

✓ can also use Scanner class

✓ loop to read till end of file

✓ ✓ Split into tokens can also use Scanner class

✓ ✓

✓

✓ reading each line correctly in loop

✓ correct method header

✓ create route object correctly
0 – if alternative constructor created and used

✓ **Efficiency mark**: File is never reread – only stop array used. No other public method created in Route/Stop class to do any part of this method.

Any way to fix array to correct length
(2 marks – see each alternative code for mark allocation ticks)
alternative 1: count first then create exact array then copy
alternative 2: create array of 100 and reduce size after fill

```
// Alternative 1: create exact array then load

int count = 0;
for (int i = 0; i < stopCount; i++) {
    if (allStops[i].isPartOfRoute(inCode))
    {
        count++;
    }
}

Stop[] arr = new Stop[count];

count = 0;

for (int i = 0; i < stopCount; i++) {
    if (allStops[i].isPartOfRoute(inCode))
    {
        arr[count] = allStops[i];
        count++;
    }
}
// End of Alternative 1


// Alternative 2: Create same size array as
// allStops – copy over and then shorten arr

Stop[] tempArr = new Stop[allStops.length];
int count = 0;
for (int i = 0; i < stopCount; i++) {
    if (allStops[i].isPartOfRoute(inCode))
    {
        tempArr[count] = allStops[i];
        count++;
    }
}

Stop[] arr = new Stop[count];
System.arraycopy(tempArr, 0, arr, 0, count);
// or use for loop to copy to actual array

// End of Alternative 2


r.setStops(arr);

return r;

}
```

✓✓ A1 Way to fixed length of array (count first)

✓ A1: create new stop array

✓ A1: loop to go through stops correct

✓ A1: check and add if Stop is part of route

✓ A2: create new stop array with sufficient length

✓ A2: loop to go through stops correct

✓ A2: check and add if Stop is part of route

✓ A2: create new stop array

✓ A2: Fixed array size by copy over to correct size array

✓ set array (can also be before load loop)

✓

```
// Q7.1 - (6)
    public String workOutStopPoints(Route r1, Route r2)
    {
      // Marks allocated according to objectives met:
        // 1 – O1: loop to go through route 1 stops
        // 1 – O2: test if it also belongs to route 2
        // 1 – O3: Efficiency – no file re-reading, no other method created
        // 1 – O4: string concatenation done correctly
        // 1 – O5: correct return (0 if there is return but not correct)

      String toret = "";

      // alternative 1: working with the two routes

      int count = 0;
      Stop s = r1.getStopAt(count);
      // s not necessary - can work directly with r1.getStopAt(count);

      while (s != null)
      {
         if (s.isPartOfRoute(r2.getRouteCode()))
         {
            toret += s + "\n";
         }
         count++;
         s = r1.getStopAt(count);
      }

      return toret;

      // End of Alternative 1

      // alternative 2: working with the allStops array
      for (int i = 0; i < stopCount; i++)
      {
         if (allStops[i].isPartOfRoute(r1.getRouteCode())
             &&
             allStops[i].isPartOfRoute(r2.getRouteCode()))
         {
             toret += allStops[i] + "\n";
         }
      }

      return toret;

      // End of Alternative 2
   }

}
```

✓ correct public method header: any name, must be public, takes in two Routes object, returns String correctly

✓ A1: O1 loop to go through r1 stops correctly

✓ A1:O2 test to see if stop is part of r2

✓ A1: O3 no file re-read, no new public method used

✓ A1: O4 String concatenation done correctly

✓ A1: O5 correct return

✓ A2: O1 loop to go through r1 stops

✓ A2:O2 test to see if stop is part of r2

✓ A2: O4 String concatenation done correctly

✓ A2: O3 no file re-read, no new public method used

✓ A2: O5 correct return

QUESTIONS 6 AND 7.2: TOURUI CLASS

```
// Q6.1 - (1)
public class TourUI {

    public static void main(String[] args) {

        // Q6.2 - (1)
        TourManager sm = new TourManager("data.txt");


        // Q6.3 - (2)
        Route routeR = sm.getRouteWithCode('R', true);
        Route routeY = sm.getRouteWithCode('Y', false);


        // Q6.4 (1) – print both routes
        System.out.println(routeR);
        System.out.println(routeY);


        // Q7.2 - (1)
        System.out.println(sm.workOutCommonStops (routeR, routeY));

    }

}
```

DELPHI SOLUTION

QUESTION 2: STOP CLASS

```
unit uStop;

interface
  uses SysUtils;

  // Q2.1 - (1)  ✓
  type TStop = class

    // Q2.2 – (3)
    private
      stopName : string;
      routeCodes : string;
      stopType : integer;

    // Q2.3 – (3)
    public
      const
        STOPTYPE_CAFE = 1;
        STOPTYPE_SHELTER = 2;
        STOPTYPE_EXPRESS = 3;
        STOPTYPE_OTHER = 4;

      constructor Create(inStopName , inRouteCodes : string;
                         inStopType : integer);
      function getStopTypeName() : string  ;
      function isPartOfRoute(r: char) : boolean;
      function toString() : string;
  end;

implementation
   // Q2.4 – (7)                ✓
   constructor TStop.Create(inStopName , inRouteCodes : string;
                            inStopType : integer);
   begin
     stopName := inStopName;
     routeCodes := inRouteCodes;

     if (inStopType = STOPTYPE_CAFE) or
         (inStopType = STOPTYPE_SHELTER) or
         (inStopType = STOPTYPE_EXPRESS) then
     begin
       stopType := inStopType;
       // also accept if set each one individually
     end
     else
     begin
       stopType := STOPTYPE_OTHER
     end;
   end;
```

✓ Private
✓ Correct Type
✓ Named as asked

✓ Final/const
✓ Named correctly
✓ Int values correct

✓ Properties set correctly

✓ Comparison correct
✓ Check against constants
✓ Use OR correctly

✓ Set only if valid

✓ otherwise set to TYPE other (not literal int)

```
// Q2.5 – (3)
function TStop.getStopTypeName() : string ;
begin
 case stopType of            // if/else if/else acceptable
   STOPTYPE_CAFE : Result := 'cafe';
   STOPTYPE_EXPRESS : Result := 'express';
   STOPTYPE_SHELTER : Result := 'shelter';
   else Result := 'other';
 end;
end;
```

✓ return correct string based on constants

```
// Q2.6 – (3)
function TStop.isPartOfRoute(r: char) : boolean;
begin
  Result := Pos(r, routeCodes) > 0;
```

✓ A1 ✓ A1

```
  // use if statement
  if Pos(r, routeCodes) > 0 then
  begin
     Result := true
  end
  else
  begin
     Result := false
  end;
end;
```

✓ A2 ✓ A2

```
// Q2.7 – (3)
function TStop.toString() : string;
begin
  Result := getStopTypeName() + #9 + stopName
end;
end.
```

✓ all content included
✓ correct format

Questions 3 and 5 : Route Class

```
unit uRoute;

interface
  uses uStop, SysUtils;

  type StopArray = array of TStop;

  // Q3.1 – (3)
  type TRoute = class
    private
      routeCode : char;
      isCircular : boolean;
      stops : StopArray;

    public
      constructor Create(inRouteCode : char; inIsCircular : boolean);
```

✓ Private
✓ Correct Type

```
      procedure setStops(inStops : StopArray);
      function getRouteCode() : char;
      function getStopAt(num : integer) : TStop;
      function toString() : string;
  end;

implementation
  // Q3.2 – (2)                                    ✓ header
  constructor TRoute.Create(inRouteCode: Char; inIsCircular: boolean);
  begin
    routeCode := inRouteCode;
    isCircular := inIsCircular;          ✓ set correctly
  end;

  // Q3.3 – (2)
  procedure TRoute.setStops(inStops : StopArray);     ✓ header
  begin
    stops := inStops; ✓ set correctly
  end;

  // Q3.4 – (1)
  function TRoute.getRouteCode() : char;
  begin
    Result := routeCode;                                ✓
  end;

  // Q3.5 – (5)                                  ✓
  function TRoute.getStopAt(num: integer) : TStop;
  begin       ✓ A1                  ✓ A1
    if (num >= 0) and (num < Length(stops)) then
    begin
      Result := stops[num]  ✓ A1 - return object
    end
    else
    begin
      Result := nil     ✓ A1 - return nil for not valid
    end;
    // also accept if argument starts from 1 as long as slot
    //processing is correct and Q7.1 also matches
    {             ✓ A2              ✓ A2
      if (num >= 1) and (num <= Length(stops)) then
      begin
        Result := stops[num-1]  ✓ A2 - return object
      end
      else
      begin
        Result := nil  ✓ A2 - return object
      end;
    }
  end;

  // Q5 – (9)
  function TRoute.toString() : string;
  var
    i : integer;
  begin
    Result := routeCode + ' - ';    ✓
```

```
    if Length(stops) = 0 then          // also accept <= 0
    begin
      Result := Result + 'Invalid';
    end
    else
    begin


      if (isCircular) then
      begin
        Result := Result + 'Circular';
      end
      else
      begin
        Result := Result + 'Linear';
      end;

      for i:= 0 to Length(stops)-1 do
      begin
        Result := Result + #13#10 + '-> ' + IntToStr(i+1)
                + #9 + stops[i].toString();
      end;

      if (isCircular) then
      begin
        Result := Result + #13#10 + '-> 1' + #9 + stops[0].toString();
      end
      else
      begin
        for i:= Length(stops)-2 downto 0 do
        begin
          Result := Result + #13#10 + '-> ' + IntToStr(i+1) + #9
                    + stops[i].toString();
        end;
      end;
    end;
  end;
end.
```

✓✓ **Efficiency mark:**
2 – no route processing for invalid stop
1 – invalid check is done, but there is unnecessary processing/checking of route type
0 – no invalid check done

✓ add circular / linear correctly

✓ include stops

✓ **Efficiency mark** – code to print out the first part of the array (where circular / linear arrays have common format) appears only once in the code i.e. no part of it appears in multiple locations.

✓ include first stop if circular

✓ loop to include stop in reverse from second last slot

✓ all stops returned in correct format

Questions 4 and 7.1 : TourManager Class

```
unit uTourManager;

interface
 uses uStop, uRoute, SysUtils;

  // Q4.1 – (1)
  type TStopManager = class ✓

    // Q4.2 – (3)                                    ✓ Private
    private                                          ✓ Correct Type
      allStops : array[1..100] of TStop;             ✓ Correct initial values
      stopCount : integer;
    public
      constructor Create(inFilename : string);
      function getRouteWithCode(inCode:char; inIsCircular:boolean):TRoute;
      function workOutStopPoints(r1, r2 : TRoute) : string;
  end;

implementation
  // Q4.3 – (9)                                     ✓ correct method header
  constructor TStopManager.Create(inFilename : string);
  var
    inFile : TextFile;
    line, tStopName, tRouteCodes : string;
    tStopType : integer;
  begin
    AssignFile(inFile, inFilename);   ✓
    Reset(inFile);

    while NOT EOF(infile) do      ✓ loop to read till end of file
    begin
      Readln(inFile, line);✓ reading each line correctly in loop

      sName := Copy(line, 1, Pos(',', line) -1 );
      Delete(line, 1, Pos(',', line));
                                                     ✓ ✓ Split into tokens
      sType := StrToInt(Copy(line, 1, Pos(',', line) -1 ));
      Delete(line, 1, Pos(',', line));
      rCodes := line;

      Inc(stopCount); ✓
      allStops[stopCount] := TStop.Create(sName, rCodes, sType);
                          ✓          ✓
    end;
  end;

  // Q4.4 – (10)              ✓ correct method header
  function TStopManager.getRouteWithCode(inCode: char ;
                      inIsCircular : boolean) : TRoute;
  var
    count, i : integer;
    tStops : StopArray;
  begin                                  ✓ create route object correctly
    Result := TRoute.Create(inCode, inIsCircular);
                                         0 – if alternative constructor created and used
            ✓ Efficiency mark: File is never reread – only stop array used. No other
            public method created in Route/Stop class to do any part of this method.
```



```
                ✓ create new stop array with sufficient length
  count := 0;
  setLength(tStops, 0);
  setLength(tStops, 100);

  for i := 1 to stopCount do     ✓ loop to go through stops correct
  begin
    if (allStops[i].isPartOfRoute(inCode)) then
    begin                                  ✓ check and add if Stop is part of route
      tStops[count] := allStops[i];
      inc(count);
    end;
  end;

  setLength(tStops, count);      ✓ ✓ reduce length of array correctly

        ✓                        ✓ set array
  Result.setStops(tStops);
end;

                                 ✓ correct public method header: any name, must be public,
                                 takes in two Routes object, returns String correctly
// Q7.1 – (6)
function TStopManager.workOutStopPoints(r1, r2 : TRoute) : string;

// Marks allocated according to objectives met:
      // 1 – O1: loop to go through route 1 stops
      // 1 – O2: test if it also belongs to route 2
      // 1 – O3: Efficiency – no file re-reading, no other method created
      // 1 – O4: string concatenation done correctly
      // 1 – O5: correct return (0 if there is return but not correct)

var
  // for alternative 1
  count : integer;
  // for alternative 2
  i :integer;
begin
  Result := '';
  // alternative 1:  working with the two route objects

  count := 0;
                                 ✓ A1: O1 loop to go through r1 stops correctly
  while (r1.getStopAt(count) <> nil) do
  begin                                  ✓ A1:O2 test to see if stop is part of r2
    if r1.getStopAt(count).isPartOfRoute(r2.getRouteCode()) then
    begin                                ✓ A1: O4 String concatenation done correctly
        Result := Result + r1.getStopAt(count).toString() + #13#10;
    end;
    inc(count);                  ✓ A1: O3 no file re-read, no new public method used
  end;
              ✓ A1: O5 correct return


  // alternative 2: working with the allstops array
  for i := 1 to stopCount do
  begin                                  ✓ A2: O1 loop to go through r1 stops
```

```
      if (allStops[i].isPartOfRoute(r1.getRouteCode()))
          and                                    ✓ A2:O2 test to see if stop is part of r2
         (allStops[i].isPartOfRoute(r2.getRouteCode())) then
      begin                                      ✓ A2: O4 String concatenation done correctly
         Result := Result + allStops[i].toString() + #13#10;
      end;       ✓ A2: O5 correct return
    end;
  end;                          ✓ A2: O3 no file re-read, no new public method used
end.
```

Questions 6 and 7.2 : TourUI Class

```
// Q6.1 – (1)
program TourUI;      ✓

{$APPTYPE CONSOLE}          // or create form
{$R *.res}

uses
  System.SysUtils,
  uStop in 'uStop.pas',
  uRoute in 'uRoute.pas',
  uTourManager in 'uTourManager.pas';

var
  temp : string;
  sm : TStopManager;
  routeR, routeY : TRoute;
begin
  try
    // Q6.2 – (1)
    sm := TStopManager.Create('data.txt');  ✓

    // Q6.3 - (2)
    routeR := sm.getRouteWithCode('R', true); ✓
    routeY := sm.getRouteWithCode('Y', false);✓

    // Q6.4 – (1) – print both routes
    WriteLn(routeR.toString());     ✓
    WriteLn(routeY.toString());

    // Q7.2 – (1)                   ✓
    WriteLn(sm.workOutStopPoints(routeR, routeY));

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

OUTPUT

SECTION A

QUESTION 1.1

TouristID	TouristName	Email	Hotel	DateRegistered
8	Jorrie Potten	jpotten@seattletimes.com	Mount Grace Hotel	2016-11-03
10	Johnath Nixon	jnixon@seattletimes.com	Peninsula Hotel	2016-05-13
11	Davis Eginton	degintona@seattletimes.com	Peninsula Hotel	2018-07-30

QUESTION 1.2
NO OUTPUT

QUESTION 1.3

ExcursionName	Duration
Sunrise Breakfast River Cruise	3
Township Excursion 1	2
National Art Museum Excursion 1	3

QUESTION 1.4

Hotel	NumberOfTourist
Mount Grace Hotel	5
Peninsula Hotel	6
President Hotel	3
Village Lodge	5

QUESTION 1.5

TouristName
Arron Haney
Cornall Prout
Darryl Poleykett
Davis Eginton
Eba Gillison
Eleen Yeomans
Irina Gouny
Johnath Nixon
Jorrie Potten
Kort McAndie
Marge Hengoed
Mic MacArd
Paul Buller
Rozalie Kebell
Sharl MacMenamy
Sharlene Bendall

QUESTION 1.6

Excursion Name	**ExcursionCode** *The last two digits will be different in each case because they are randomly generated.*
Sunrise Breakfast River Cruise	Sun96
Sunset River Cruise	Sun56
Morning Safari	Mor32
Afternoon Safari	Aft31
Night Safari	Nig25
Township Excursion 1	Tow36
Township Excursion 2	Tow84
Township Excursion 3	Tow82
Township Excursion 4	Tow18
Township Excursion 5	Tow25
National Art Museum Excursion 1	Nat82
National Art Museum Excursion 1	Nat70
National Art Museum Excursion 1	Nat39
Splash Marine Park Morning Excursion	Spl48
Splash Marine Park Afternoon Excursion	Spl76
Splash Marine Park Day Excursion	Spl20
National Park Day Excursion	Nat19

QUESTION 1.7

NO OUTPUT

SECTION B

FINAL OUTPUT

```
R - Circular
-> 1   cafe         Waterfront
-> 2   express      St Monicas Cathedral
-> 3   shelter      Conference Centre
-> 4   express      CC
-> 5   shelter      Market Square
-> 6   express      Jewel Africa
-> 7   shelter      Taemane National Park
-> 8   shelter      Igugu Marine Park
-> 9   express      President Hotel
-> 10  other        St Johns Road
-> 11  express      Winchester Hotel
-> 12  shelter      Soccer Stadium
-> 1   cafe         Waterfront
Y - Linear
-> 1   shelter      Market Square
-> 2   shelter      Clock Tower
-> 3   express      Idayimani Museum
-> 4   other        Grande Hotel
-> 5   shelter      SA Heritage Museum
-> 6   shelter      Apartheid Museum
-> 7   express      Fort of Idayimani
-> 6   shelter      Apartheid Museum
-> 5   shelter      SA Heritage Museum
-> 4   other        Grande Hotel
-> 3   express      Idayimani Museum
-> 2   shelter      Clock Tower
-> 1   shelter      Market Square
shelter      Market Square
```


ANNEXTURE A: ALTERNATE SOLUTION

DELPHI: Questions 3.5 and 5 : Route Class with Static Array

```
unit uRoute;

interface
  uses uStop, SysUtils;
  // Q3.1 – 1 Mark                                    ✓
  type StopArray = array[1..100] of TStop;

  type TRoute = class
    private
      routeCode : char;
      isCircular : boolean;
      stops : StopArray;
      numStopsFound : integer;  // needed for Q3.5
  ...


  // Q3.3 - 2 Marks
  procedure TRoute.setStops(inStops : StopArray);
  var
    i : integer;
  begin
    stops := inStops;
    numStopsFound := 0;

    for i := 1 to 100 do
    begin
      if stops[i] <> nil then
      begin
        Inc(numStopsFound);
      end
    end
  end;
  ...


  // Q3.5 – (5)                                       ✓
  function TRoute.getStopAt(num: integer) : TStop;
  begin      ✓                                       ✓ only if object stop number
    if (num >= 1) and (num <= numStopsFound) then     has been tracked correctly
    begin                                             (see Q3.3)
      Result := stops[num]   ✓ return object
    end
    else
    begin
      Result := nil      ✓ return nil for not valid
    end;
  end;
```

Need this loop to work out how many stops there are in this array

```
// Q5 - (9)
function TRoute.toString() : string;
var
  i : integer;
begin
  Result := routeCode + ' - ';

  if Length(stops) = 0 then          // also accept <= 0
  begin
    Result := Result + 'Invalid';
  end
  else
  begin

    if (isCircular) then
    begin
      Result := Result + 'Circular';
    end
    else
    begin
      Result := Result + 'Linear';
    end;

    for i:= 1 to numStopsFound do
    begin
      Result := Result + #13#10 + '-> ' + IntToStr(i)
              + #9 + stops[i].toString();
    end;

    if (isCircular) then
    begin
      Result := Result + #13#10 + '-> 1' + #9 + stops[1].toString();
    end
    else
    begin
      for i:= numStopsFound-1 downto 1 do
      begin
        Result := Result + #13#10 + '-> ' + IntToStr(i) + #9 +
                  stops[i].toString();
      end;
    end;
  end;
end;
end.
```

✓✓ **Efficiency mark:**
2 – no route processing for invalid stop
1 – invalid check is done, but there is unnecessary processing/checking of route type
0 – no invalid check done

✓ add circular / linear correctly

✓ include stops

✓ **Efficiency mark** – code to print out the first part of the array (where circular / linear arrays have common format) appears only once in the code i.e. no part of it appears in multiple locations.

✓ include first stop if circular

✓ loop to include stop in reverse from second last slot

✓ all stops returned in correct format

Questions 4.4 and 7.1 : TourManager Class with Static Array

```
unit uTourManager;

interface
 uses uStop, uRoute, SysUtils;

 ...

  // Q4.4 – (10)
  function TStopManager.getRouteWithCode(inCode: char ;
                      inIsCircular : boolean) : TRoute;
  var
    count, i : integer;
    tStops : StopArray;
  begin
    Result := TRoute.Create(inCode, inIsCircular);

    count := 1;

    for i := 1 to Length(tStop) do
    begin
      tStops[i] := nil;
    end;

    for i := 1 to stopCount do
    begin
      if (allStops[i].isPartOfRoute(inCode)) then
      begin
        tStops[count] := allStops[i];
        inc(count);
      end;
    end;

    Result.setStops(tStops);
  end;
```

✓ create new stop array with sufficient length – awarded on definition of StopArray = array[1..100] in the Route definition

✓ correct method header

✓ create route object correctly
0 – if alternative constructor created and used

✓ **Efficiency mark**: File is never reread – only stop array used. No other public method created in Route/Stop class to do any part of this method.

✓ ✓ to set any unused slots to nil (can also be after search

✓ loop to go through stops correct

✓ check and add if Stop is part of route

✓ ✓ set array

✓ correct public method header: any name, must be public, takes in two Routes object, returns String correctly

```
// Q7.1 – (6)
function TStopManager.workOutStopPoints(r1, r2 : TRoute) : string;

// Marks allocated according to objectives met:
      // 1 – O1: loop to go through route 1 stops
      // 1 – O2: test if it also belongs to route 2
      // 1 – O3: Efficiency – No file re-reading, No other method created
      // 1 – O4: string concatenation done correctly
      // 1 – O5: correct return (0 if there is return but not correct)

var
  // for alternative 1
  count : integer;
  // for alternative 2
  i :integer;
begin
  Result := '';
  // alternative 1:  working with the two route objects

  count := 1;
```

✓ A1: O1 loop to go through r1 stops correctly

```
  while (r1.getStopAt(count) <> nil) do
  begin
```

✓ A1:O2 test to see if stop is part of r2

```
    if r1.getStopAt(count).isPartOfRoute(r2.getRouteCode()) then
    begin
```

✓ A1: O4 String concatenation done correctly

```
        Result := Result + r1.getStopAt(count).toString() + #13#10;
    end;
    inc(count);
  end;
```

✓ A1: O3 no file re-read, no new public method used

✓ A1: O5 correct return

```
  // alternative 2: working with the allstops array
  for i := 1 to stopCount do
  begin
```

✓ A2: O1 loop to go through r1 stops

```
    if (allStops[i].isPartOfRoute(r1.getRouteCode()))
        and
       (allStops[i].isPartOfRoute(r2.getRouteCode())) then
    begin
```

✓ A2:O2 test to see if stop is part of r2

✓ A2: O4 String concatenation done correctly

```
       Result := Result + allStops[i].toString() + #13#10;
    end;
  end;
end;
end.
```

✓ A2: O5 correct return

✓ A2: O3 no file re-read, no new public method used