NATIONAL SENIOR CERTIFICATE EXAMINATION
NOVEMBER 2019

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 STUDENT
WHERE IsSenior = TRUE
ORDER BY StudentName
```

Question 1.2 (4)

```
SELECT ProjectName FROM PROJECT both SELECT and FROM correct
WHERE ProjectName
  LIKE
  '%tutor%'  % must appear before and after
              '*tutor*' for ACCESS - * must appear before and after
```

Question 1.3 (3)

```
SELECT RegistrationID,
      AccumulatedHours/NumAttended   correct fields
                                     correct division
      FROM REGISTRATION
```

Question 1.4 (4)

```
SELECT DISTINCT
SUBSTR (ProjectName,1,3) ACCESS: LEFT (ProjectName, 3)
FROM PROJECT

ALTERNATIVE:

SELECT SUBSTR (ProjectName,1,3)    ACCESS: LEFT  (ProjectName, 3)
FROM PROJECT
GROUP BY SUBSTR(ProjectName,1,3)   ACCESS: LEFT(ProjectName, 3)
```

Question 1.5 (4)

```
SELECT ProjectName, StudentName    Both Fields correct
FROM STUDENT, PROJECT
WHERE StudentLeaderID = StudentID
```

Question 1.6 (5)

```
SELECT StudentID, SUM (AccumulatedHours)
  AS TOTALHOURS

FROM REGISTRATION
GROUP BY StudentID
```

Question 1.7 (4)

```
SELECT StudentName FROM STUDENT
WHERE StudentID
  NOT IN  Both NOT and IN
  (SELECT StudentID
      FROM REGISTRATION
   )

ALTERNATIVE:

SELECT StudentName FROM STUDENT
LEFT JOIN REGISTRATION
ON STUDENT.StudentID = REGISTRATION.StudentID
WHERE RegistrationID
      IS NULL
```

Question 1.8 (4)

```
SELECT ProjectName FROM PROJECT
WHERE DateStarted =
( SELECT MAX (DateStarted)
   FROM PROJECT
)
```

Question 1.9 (8)

```
SELECT  StudentName, Count(*) AS NumProjects
FROM REGISTRATION, STUDENT  Both tables
WHERE STUDENT.StudentID=REGISTRATION.StudentID
GROUP BY StudentName
HAVING COUNT(*) >= 2
```

JAVA SOLUTION

QUESTION 2 STUDENT CLASS

```
// Question 2.1 - 4
public class Student              // Class header
{
    private String fullName;      // all attributes private
    private int grade;            // all correctly named
    private String interest;      // correct types
    private double hours;

// Question 2.2 - 3
    public Student(String inFn, int inG, String inI, double inH)
                                  // correct header and parameter names
    {
        fullName = inFn; // fields set to parameters
        grade = inG;
        interest = inI;
        hours = inH;
    }
```

```
// Question 2.3 - 2
    public int getGrade() // correct headers for all three methods
    {
        return grade;
    }
    public String getInterest() // correct returns for all three methods
    {
        return interest;
    }
    public double getHours()
    {
        return hours;
    }

// Question 2.4 - 4
    private String alterName() // correct header
    {
        String temp = "";
        Scanner sc = new Scanner(fullName);
        String fName = sc.next(); // isolate surname
        String sName = sc.next(); // isolate firstname & initial
        return sName + ", " + fName.charAt(0);
                             // combine in the correct format
    }

// Question 2.5 - 3
    @Override
    public String toString() // correct header
    {
        return alterName() + "\t" + grade + "\t" + interest + "\t" + hours;
        // call alterName()
        // correct format with tabs
    }
```

QUESTION 3 GRADE CLASS

```
// Question 3.1 - 2
public class Grade // correct name
{
    private int grade; // correct fields grade and total
    private double total;

// Question 3.2 – 2
    private final int LIMIT = 20; // correct values assigned
    private final int BONUS = 10; // constant fields declared


// Question 3.3 - 4
    public Grade(int inG, double inT)
    {
        grade = inG;                // assign grade field
        if (inT > LIMIT)            // compare inT to LiMIT
            total = inT + BONUS;    // add BONUS to total when > limit
        else total = inT;           // otherwise assign inT to total
    }
// Question 3.4 - 2
    public double getTotal() // correct accessor method
    {
        return total;
    }

    public void setTotal(double inT) // correct mutator method
    {
        total = inT;
    }

// Question 3.5 - 2
    @Override
    public String toString()        // correct header
    {
        return "Grade:" + grade + "    total hours " + total;
                                    // fields correctly combined
    }
}
```

QUESTION 4, 6 & 7 PROJECT CLASS

```
// Question 4.1 - 2
public class Project // correct header
{
    private String name; // fields name and max correctly declared
    private int max;

// Question 4.2 - 4
    private Student sArr[] = new Student[50]; // correct object name
                                              // 50 elements declared
    private int sCount = 0;                  // sCount correctly declared
    private Grade gArr[] = new Grade[5]; // gArr correctly declared

// Question 4.3 - 8
    public Project(String inN, String inC, int inM)
    {
        name = inN; // fields name and max correctly assigned
        max = inM;
        try
        {
            Scanner scFile = new Scanner(new FileReader("Students.txt"));
                                              // file opened for reading
            String nm, i;
            int g;
            double h;
            while (scFile.hasNextLine())   // loop to read file
            {
                String line = scFile.nextLine();

                Scanner scLine = new Scanner(line).useDelimiter(",");

                nm = scLine.next();
                g = scLine.nextInt();
                i = scLine.next();
                h = scLine.nextDouble();
                                         // extract fields

                if (i.equalsIgnoreCase(inC) || i.equals("Both"))
                                        // check for "Both" and category type
                {
                    sArr[sCount] = new Student(nm, g, i, h);
                                    // instatiate Student array object
                    sCount++;       // increment sCount
                }
            }
            scFile.close();
        } catch (FileNotFoundException ex)
        {
            System.out.println("File not found");
        }
    }

// Question 4.4 - 4
    public String toString()
    {
        String temp = "Name:\t" + name + "\nMaximum:" + "\t" + max + "\n";
        // correct header information and format

        for (int i = 0; i < sCount; i++)   // loop through array
        {
            temp += sArr[i] + "\n";        // combine into a string
        }
        return temp;                       // return combined string
    }
}

// Question 4.5 - 6
    public void sort()
    {
        for (int i = 0; i < sCount - 1; i++)      // correct outer loop
        {
            for (int j = i + 1; j < sCount; j++)  // correct inner loop
            {
                if (sArr[i].getGrade() > sArr[j].getGrade())
                                                 // correct if statement
                                               // sorts in correct order
                {
                    Student temp = sArr[i];
                    sArr[i] = sArr[j];
                    sArr[j] = temp;     // correct swop
                }
            }
        }
    }
```

```
// Question 6 - 13
//4 marks for a successful delete - the delete does not need to be in a
separate method
    private void deleteStudent(int inP)
    {
        sArr[inP] = sArr[sCount - 1];
                    // correct code to delete item
        sort();
        sCount--;
        // decrement sCount
    }

//ALTERNATE deleteStudent
    private void deleteStudent(int inP)
    {
        for (int i = inP; i < sCount; i++)
        {
                sArr[i] = sArr[i+1];   // correct code to delete item

        sCount--;             // decrement sCount

    }

// 9 Marks to process array
    public String correctNumbers()
    {
        if (sCount > max)     // check if items need to be deleted
        {
            int amountToDelete = sCount - max; // determine amount to delete
            String deletedList = "Students removed:\n";
            for (int i = 0; i < amountToDelete; i++)  // loop through array
            {
                int item = (int) (Math.random() * sCount);
                                      // generate random number within range
                deletedList += sArr[item] + "\n";
                                      // create string with deleted items
                deleteStudent(item);
                         // call delete method OR delete code is placed here
            }
            return deletedList + "\nRemaining students:\n" + toString();
           // remaining string created both strings correct with headers
    }

// Question 7 - 9
//6 marks to instantiate GradeArray gArr
    public String createGradeArray(String inH) // correct header
    {
        int grade;
        double hours;

        Scanner scLine = new Scanner(inH).useDelimiter(";");
        while (scLine.hasNext())
        {

            grade = scLine.nextInt();
            hours = scLine.nextDouble(); // fields extracted correctly
```

```
                gArr[grade - 8] = new Grade(grade, hours);
                                         // gArr correctly instantiated
            }
            scLine.close();
            return displayAllGradeArray(); // Return a string
        }

//3 marks to create a string of gArr – this does not need to be a separate
method
        private String displayAllGradeArray()
        {
            String tempSt = "\nGrade Totals:\n"; // Correct heading

            for (int i = 0; i < 5; i++)     // loop to process Grade array
            {
                tempSt += gArr[i] + "\n"; // objects correctly combined

            }
            return tempSt;
        }
```

QUESTION 5 & 8 PROJECTUI CLASS

```
// Question 5.1 - 1
public class ProjectUI  // correct header
{

    public static void main(String[] args)
    {
       // Question 5.2 - 1
       Project clothes = new Project("Collect old clothes","Indoor",12);
       // project correctly instantiated

       // Question 5.3 - 2
       clothes.sort();
       System.out.println(clothes);
       // sort method called and object displayed

       // Question 8.1 - 1
       System.out.println(clothes.correctNumbers());
       // call correctNumbers correctly

       // Question 8.2 - 1
       System.out.println
(clothes.createGradeArray("9;53;12;13;8;72;11;90;10;34"));
       // call createGradeArray correctly
    }

}
```

DELPHI SOLUTION

QUESTION 2 STUDENT CLASS

```
unit uStudent;

interface
  uses SysUtils;

// Question 2.1 - 4
type TStudent = class         // Class header
    private                   // all attributes private
      fullName : string;      // all correctly named
      grade : integer;        // correct types
      interest : string;
      hours : double;
    public
      constructor Create(inFn: string; inG: integer; inI: string; inH: double);
      function getGrade() : integer  ;
      function getInterest(): string;
      function getHours() : double;
      function alterName() : string;
      function toString() : string;
  end;

implementation

// Question 2.2 - 3
  constructor TStudent.Create(inFn: string; inG: integer; inI: string;
       inH: double);      // correct header and parameter names
  begin
    fullName := inFn;     // fields set to parameters
    grade := inG;
    interest := inI;
    hours := inH;
  end;


// Question 2.3 - 2
  function TStudent.getGrade() : integer;      // correct headers for all three
methods
  begin
    Result := grade;
  end;

  function TStudent.getInterest() : string;    // correct returns for all three
methods

  begin
    Result := interest;
  end;

 function TStudent.getHours() : double;
  begin
    Result := hours;
  end;
```

```
// Question 2.4 - 4
 function TStudent.alterName() : string;   // correct header
 var
   surname, initial: string;
 begin
    initial := fullName[1];  // isolate firstname & initial
    surname := Copy(fullName, Pos(' ', fullName) +1, Length(fullName ));
                            // isolate surname
    Result := surname + ', ' + initial;  // combine in the correct format
  end;

// Question 2.5 - 3
  function TStudent.toString() : string;   // correct header
  var i : integer;
  begin
    Result := alterName() + #9 + IntToStr(grade) + #9 + interest
            + #9 + floattostr(hours);
         // call alterName()
         // correct format with tabs
  end;
end.
```

QUESTION 3 GRADE CLASS

```
unit uGrade;

interface
  uses SysUtils;

// Question 3.1 - 2
  type TGrade = class      // correct name
    private
      grade : integer;     // correct fields grade and total
      total : double;

      // Question 3.2 - 2
      const
        LIMIT = 20;        // correct values assigned
        BONUS = 10;        // constant fields declared
    public
      constructor Create(inG: integer; inT: double);
      function getTotal() : double  ;
      procedure setTotal(inT: double);
      function toString() : string;
  end;

implementation

  // Question 3.3 - 4
  constructor TGrade.Create(inG: integer; inT: double);
  begin
    grade := inG;                 // assign grade field

    if (inT > LIMIT) then         // compare inT to LiMIT
    begin
      total := inT + BONUS;       // add BONUS to total when > limit
    end
    else
    begin
      total := inT;                 // otherwise assign inT to total
    end;
  end;

// Question 3.4 - 2
  function TGrade.getTotal() : double;
  begin                          // correct accessor method
    Result := total;
  end;

  procedure TGrade.setTotal( inT: double); // correct mutator method
  begin
    total := inT;
  end;

// Question 3.5 - 2
  function TGrade.toString() : string;    // correct header
  var i : integer;
  begin
    Result := 'Grade: ' + IntToStr(grade) + '   total hours '
            + floattostr(total);    // fields correctly combined
  end;
end.
```

QUESTION 4, 6 & 7 PROJECT CLASS

```
unit uProject;

interface
uses
    SysUtils, uStudent, uGrade;

// Question 4.1 - 2
  type TProject = class       // correct header
    private
      name : string;          // fields name and max correctly declared
      max : integer;
      function displayAllGradeArray() : string  ;
    public

      // Question 4.2 - 4
      sArr : array[1..50] of TStudent;    // correct object name
                                          // 50 elements declared
      sCount : integer;                   // sCount correctly declared
      gArr : array[1..5] of TGrade;       // gArr correctly declared
      constructor Create(inN, inC : string; inM: integer);
      function toString() : string;
      procedure sort();
      procedure deleteStudent(inP : integer);
      function correctNumbers() : string;
      function createGradeArray(inH : string) : string;
  end;
implementation
```

```
// Question 4.3 - 8
  constructor TProject.Create(inN, inC : string; inM: integer);
  var
    inFile : TextFile;     // fields name and max correctly assigned
    line : string;

    nm,i : string;
    g : integer;
    h : double;
  begin
    name := inN;
    max := inM;
    sCount := 0;

    AssignFile(inFile, 'Students.txt');  // file opened for reading
    Reset(inFile);

    while NOT EOF(inFile) do             // loop to read file
    begin
      Readln(inFile, line);

      nm := Copy(line, 1, Pos(',', line) -1 );
      Delete(line, 1, Pos(',', line));
      g := StrToInt(Copy(line, 1, Pos(',', line) -1));
      Delete(line, 1, Pos(',', line));
      i := Copy(line, 1, Pos(',', line) -1 );
      Delete(line, 1, Pos(',', line));
      h := strtofloat(line);
                                          // extract fields

      if (CompareText(inC, i)=0) or (i='Both') then
                                        // check for "Both" and category type
      begin
        sCount := sCount + 1;      // increment sCount
        sArr[sCount] := TStudent.Create(nm, g, i, h);
                                   // instatiate Student array object
      end;

    end;

  end;

// Question 4.4 - 4
  function TProject.toString() : string;
  var i : integer;
  begin
    Result := 'Name: ' + #9 + name + #10#13 + 'Maximum:' + #9 + IntToStr(max)
           + #13#10;  // correct header information and format
    for i:= 1 to sCount do                 // loop through array
      Result := Result + sArr[i].toString() + #13#10;
                                           // combine into a string
                                           // return combined string
  end;

// Question 4.5 - 6
  procedure TProject.sort() ;
  var
```

```
    i, j : integer;
    temp : TStudent;
  begin
    for i:= 1 to sCount do         // correct outer loop
      for j:= 1 to sCount do       // correct inner loop
      begin
        if (sArr[i].getGrade() < sArr[j].getGrade()) then
        begin                      // correct if statement
                                   // sorts in correct order
          temp := sArr[i];
          sArr[i] := sArr[j];
          sArr[j] := temp;         // correct swop
        end;
      end;
  end;

// Question 6 - 13
//4 marks for a successful delete - the delete does not need to be in a
separate method
  procedure TProject.deleteStudent(inP: Integer);
  begin
    sArr[inP] := sArr[sCount - 1];
    sort();                  // correct code to delete item

    sCount := sCount -1;     // decrement sCount
  end;

//ALTERNATE deleteStudent
  procedure TProject.deleteStudent(inP: Integer);
  begin
    for i:= inP to sCount do
        sArr[i] := sArr[i+1];      // correct code to delete item
    sCount := sCount -1;           // decrement sCount

  end;

// 9 Marks to process array
  function TProject.correctNumbers() : string;
  var
    amountToDelete, i, randPos : integer;
    deletedList : string;
  begin
    if (sCount > max) then               // check if items need to be deleted
    begin
      amountToDelete := sCount - max;    // determine amount to delete
      deletedList := 'Students removed:' + #13#10;

      for  i := 1 to amountToDelete do      // loop through array

      begin
        randPos := Random(sCount)+1;
                                    // generate random number within range
        deletedList := deletedList + sArr[randPos].toString() + #13#10;
                                        // create string with deleted items
        deleteStudent(randPos);
                         // call delete method OR delete code is placed here
      end;
      Result := deletedList + #13#10 + 'Remaining students:' +  #13#10
              + toString();
      // remaining string created both strings correct with headers

    end;

  end;
```

// Question 7 – 9
//6 marks to instantiate GradeArray gArr

```
  function TProject.createGradeArray(inH : string) : string;
                                                  // correct header
  var
    grade : integer;
    hours : double;
  begin
    while Pos(';', inH) > 0 do
    begin
      grade := StrToInt(Copy(inH, 1, Pos(';', inH) -1 ));
      Delete(inH, 1, Pos(';', inH));
      if (Pos(';', inH) <> 0) then
      begin
        hours := StrToFloat(Copy(inH, 1, Pos(';', inH) -1));
        Delete(inH, 1, Pos(';', inH));
      end
      else hours:= StrToFloat(inH);
                                        // fields extracted correctly

      gArr[grade-7] := TGrade.Create(grade, hours);
                                        // gArr correctly instantiated
    end;
    Result := displayAllGradeArray; // Return a string
  end;
```

//3 marks to create a string of gArr – this does not need to be a separate method function TProject.displayAllGradeArray() : string;

```
  var
    i : integer;
  begin
    Result := #13#10 + 'Grade Totals:' + #13#10;  // Correct heading

    for i:=1 to 5 do                         // loop to process Grade array
      Result := Result + gArr[i].toString() + #13#10;
                                       // objects correctly combined
  end;

end.
```

QUESTION 5 & 8 PROJECTUI CLASS

```
// Question 5.1 - 1
program ProjectUI;          // correct header

{$APPTYPE CONSOLE}

{$R *.res}

uses
  System.SysUtils,
  uStudent in 'uStudent.pas',
  uProject in 'uProject.pas',
  uGrade in 'uGrade.pas';

var
  clothes : TProject;
  temp : string;
begin
  try

    // Question 5.2 - 1
    clothes := TProject.Create('Collect old clothes', 'Indoor', 12);
    // project correctly instantiated

    // Question 5.3 - 2
    clothes.sort();
    WriteLn(clothes.toString());
    // sort method called and object displayed

    // Question 8.1 - 1
    WriteLn(clothes.correctNumbers());
    // call correctNumbers correctly

    // Question 8.2 – 2
    Writeln(clothes.createGradeArray (‘9;53;12;13;8;72;11;90;10;34'));
    // call createGradeArray correctly

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

OUTPUT

SECTION A

QUESTION 1.1

StudentID	StudentName	IsSenior
53	Amy Radebe	TRUE
30	Andrea Badenhorst	TRUE
54	Brendan Smit	TRUE
1	Jacob Ncube	TRUE
25	Joshua Jacobs	TRUE
22	Julia Hudson	TRUE
3	Karabo Mlangeni	TRUE
47	Kendal Buys	TRUE
12	Kenneth Motala	TRUE
17	Kobus Venter	TRUE
37	Laetitia Adams	TRUE
38	Lesego Semenya	TRUE
11	Michael Stemmet	TRUE
49	Mikyle Sithole	TRUE
43	Mthokozisi Kumalo	TRUE
39	Nina Ntsimango	TRUE
9	Patricia Davids	TRUE
16	Penny Mbele	TRUE
44	Prince Dube	TRUE
20	Rethabile Mokone	TRUE
46	Sego Dlamini	TRUE
27	Somizi Baloyi	TRUE
8	Steven Govender	TRUE
51	Tasneem Morkel	TRUE
33	Vicki de Beer	TRUE
41	Wian Oosthuizen	TRUE
18	Wiseman Legodi	TRUE

QUESTION 1.2

ProjectName
SAT English tutoring program
FRI Mathematics tutorials

QUESTION 1.3 *Data may be formatted differently on your computer*

RegistrationID	Expr1001
1	1.0833333333333333
2	0.8928571428571429
3	0.8
4	0.7857142857142857
5	0.84375
6	1.09375
7	0.7083333333333334
8	0.9038461538461539
9	0.925
10	0.9583333333333334
11	0.9318181818181818
12	1.0666666666666667
13	1.125
14	0.8125
15	1.0
16	1.0

Continue on the next page

RegistrationID	AveHours
17	0.875
18	0.5
19	0.8333333333333334
20	1.125
21	0.8846153846153846
22	1.5
23	0.9318181818181818
24	1.5
25	0.9090909090909091
26	0.5
27	0.8636363636363636
28	1.0
29	0.5
30	1.5
31	0.5
32	0.75
33	0.8
34	0.95
35	0.5
36	1.5
37	0.8
38	1.5
39	1.0555555555555556
40	0.7
41	1.0
42	1.5
43	0.8
44	0.9444444444444444
45	0.75
46	1.0
47	0.5
48	1.0
49	0.6
50	1.0
51	1.5
52	0.5
53	0.8333333333333334
54	0.5
55	1.5
56	1.2
57	0.9444444444444444
58	0.9
59	0.875
60	0.875
61	0.9166666666666666
62	0.8125
63	0.9375
64	0.96875
65	1.5
66	0.5
67	0.9285714285714286
68	0.9285714285714286
69	1.0
70	0.8125
71	0.5
72	0.5
73	1.5

QUESTION 1.4

Day
FRI
MON
SAT
THU
TUE
WED

QUESTION 1.5

ProjectName	StudentName
THU Sandwich feeding scheme	Jacob Ncube
SAT English tutoring programme	Karabo Mlangeni
SAT River clean up	Penny Mbele
FRI Mathematics tutorials	Kenneth Motala
TUE Knitting for moms	Patricia Davids
MON Old Age Home Visit	Steven Govender
WED Recycling Programme	Wiseman Legodi
FRI Textbook Collection	Joshua Jacobs

QUESTION 1.6

StudentID	TotalHours
1	13,5
2	16,25
3	8,5
4	17,5
5	9,25
6	13,5
7	10,5
8	7
9	5,5
10	11,5
11	19,5
12	10,25
13	14
14	16,5
15	14,5
16	10
17	18,25
18	6,5
19	7
20	3,75
21	10
22	12,5
23	11,75
24	10
25	15
26	11,5
27	6
28	4
29	17,5
30	11
31	14,5
32	7,5
33	15
35	15
36	6
37	16
38	8

StudentID	TotalHours
39	9,5
40	8,5
41	10
42	15
43	10,75
44	6
45	8
46	13
47	6
49	10,5
50	9
51	12,5
52	4,5
53	8
54	8,25
55	13

QUESTION 1.7

StudentName
Khaya Mokoena
Mishka Hassen

QUESTION 1.8

ProjectName
TUE Knitting for moms
FRI Textbook Collection

QUESTION 1.9

StudentName	NumProjects
Bhule Mbasa	2
Blessing Mkhize	2
Conrad Snyman	2
Frans Theron	2
Heinriche Pretorius	2
Joshua Jacobs	2
Kobus Venter	2
Laetitia Adams	2
Mahmood Chetty	3
Mary-anne Muir	2
Michael Stemmet	2
Mikyle Sithole	2
Mthokozisi Kumalo	2
Nikita van Wyk	2
Nina Ntsimango	2
Patience Madonsela	2
Sego Dlamini	2
Tasneem Morkel	2
Vicki de Beer	2

SECTION B

FINAL OUTPUT

```
Name:  Collect old clothes
Maximum:     12
Pettie, F    8      Indoor       2.5
Maler, S     8      Both         0.0
Honiford, E  8      Indoor       5.5
Doyle, J     9      Both         4.0
Boyder, L    9      Both         9.5
Delaney, B   9      Indoor       3.0
Leaby, T     9      Both         7.0
Shorts, H    9      Both         12.0
Rabey, C     10     Indoor       4.0
Janson, C    10     Both         7.0
Scotty, M    10     Indoor       1.0
Leaby, G     10     Indoor       3.0
Morvel, M    11     Both         1.0
Heriot, L    11     Indoor       3.5
McCalum, A   11     Both         5.0
Monahan, B   12     Indoor       5.0


Students removed:
// this list will vary depending on which students were randomly selected
Leaby, T     9      Both         7.0
Shorts, H    9      Both         12.0
Scotty, M    10     Indoor       1.0
Pettie, F    8      Indoor       2.5

Remaining students:
Name:  Collect old clothes
Maximum:     12
// this list will vary depending on which students were randomly deleted
Maler, S     8      Both         0.0
Honiford, E  8      Indoor       5.5
Doyle, J     9      Both         4.0
Boyder, L    9      Both         9.5
Delaney, B   9      Indoor       3.0
Rabey, C     10     Indoor       4.0
Janson, C    10     Both         7.0
Leaby, G     10     Indoor       3.0
Morvel, M    11     Both         1.0
Heriot, L    11     Indoor       3.5
McCalum, A   11     Both         5.0
Monahan, B   12     Indoor       5.0


Grade Totals:
Grade:8     total hours 82.0
Grade:9     total hours 63.0
Grade:10     total hours 44.0
Grade:11     total hours 100.0
Grade:12     total hours 13.0
```