BETEC NATIONAL IT PRACTITIONERS

SOFTWARE DEVELOPMENT

IVA

UNIT 8

PROGRAMMING CONCEPTS and PRACTICE

Tournament Organising System

STUDENT: Sophie Greene

DATE: 21ST,June 2007

Table of Contents

1 Introduction

The main purpose of this Assignment is to investigate, plan, design, implement, test and document a tournament organising system. The system will produce games schedule, will allow the tournament organisers to enter the results of the matches and finally the programme will decide who is the winner based on the results entered.

2 Planning and Analysis (Task1)

The first step of this project is to investigate the client requirements, system requirements and the programmer requirements.

2.1 Programmer's Requirements

To develop the required system, the programmer will need hardware, software and time. The specifications are listed below

2.1.1 Hardware Requirements

Item	Specification
PC	300 megahertz or higher processor clock speed recommended; 233 MHz minimum required (single or dual processor system);* Intel Pentium/Celeron family, or AMD K6/Athlon/Duron family, or compatible processor recommended
RAM	128 megabytes (MB) of RAM or higher recommended (64 MB minimum supported; may limit performance and some features)
Hard Disk	1.5 gigabytes (GB) of available hard disk space*
Monitor	Super VGA (800 x 600) or higher-resolution video adapter and monitor
Keyboard and Mouse	Keyboard and Microsoft Mouse or compatible pointing device

Table 1

2.1.2 Software Requirements

Item	Specification
OS	Windows 2000, or Windows XP
software	Visual Basic 6.0, Microsoft Office

Table 2

2.1.3 Action Plan (Time)

Table 1 shows my action plan, which is an estimation of the dates on which the project stages will be completed.

Starts: 19th February 2007								Ends: 21st May 2007		
Task	**Wk1**	**Wk2**	**Wk3**	**Wk4**	**Wk5**	**Wk6**	**Wk6**	**Wk8**	**Wk9**	**Wk10**
Analysis										
user requirements										
system requirements										
Inputs/Outputs/Processes										
Design										
Structure Diagrams										
Implementation										
Testing										
Evaluation										

Table 3 (Action Plan)

2.2 Client's Requirements

Salchester Primary School requires the following specification in the proposed system:

1. The programme should prepare a list of matches for the game's competition, bearing in mind the following rules:
    - The games that will be played are Snap, Cribbage, Spillikins and Junior Scrabble.
    - The Yellow and Green houses compete against each other in a series of matches.
    - Each match is 'best of three' single games of one type of games.
    - Students from the same class can not play each other.
    - A student can play another student only once.
    - Each student can play each game only once.
2. The programme should identify the winner of each match on the schedule.
3. The programme should calculate the points awarded to each house.
4. Appropriate colour scheme should be used in the GUI (Graphical User Interface) of the programme; the colours of the Yellow and Green houses.

5. The programme's user interface should be designed in an easy to use way; so that any member of school's staff can deal with it without any difficulty.
6. The programme should be thoroughly tested as no mistakes are tolerated on the day of the event.
7. The list of matches can only be available to students on the day of the event.
8. Once all the results of the matches are entered, a greeting window with an image of a cup decorated in the winner house colour should be displayed.

2.3 System Requirements

The programme should be designed in a way that fulfils the client's requirement. In the following subsections, a description of the user interface, inputs, processes, outputs and data storage is provided. The description is based on the client's requirements. The optimal and minimum hardware and software needed to run the programme will be illustrated.

2.3.1 User Interface

The user interface will be designed using Visual Basic 6.0. It will contain many forms (windows).

1. A login form will be provided to prevent unauthorised access to the programme
    - Two text boxes will be provided to allow the user to enter their username and password.
    - An OK command button will be provided so that the user confirms the username and password entered
    - A CANCEL command button will be provided to allow the user to cancel the login.
2. One form will be designed to allow the user to enter the number of matches for each game.
    - Four Combo boxes will be provided to enter the number of matches for each game.
    - A command button will be provided with appropriate caption. When clicked, it will generate the games matches' schedule.
    - A command button will be provided to exit the programme.

3. Another form will be designed to display the matches generated.
    - Labels will be used to display the opponents in each match.
    - Text boxes will be provided next to each label to allow the user to enter each opponent's result.
    - A command button will be provided with appropriate caption. When clicked, it will calculate the score of each house, and hence decide which house is the winner.
    - A command button will be provided to exit the programme.
4. Another form will be designed to greet the winner.
    - It will contain an image of the tournament cup
    - The colours used on the form will reflect the colour of the house which won the competition.

2.3.2 Inputs

- List of students (stored in an Access file)
- Class to which each student belongs
- House to which each student belongs
- The result of each student in each game

2.3.3 Outputs

- Score of each house
- The winner

2.3.4 Processing

- Generate the schedule,
- Calculate each house score
- Determine the winner

2.3.5 Data Storage

- Integer data type variables will be used to count the result of each house.
- String data type variable will be used to hold the names of students
- A list of students, their houses and classes will be stored in a table in a Microsoft Access file. This will be used by the programme to read each student's information. It will also help in fulfilling the rules of the game, listed in the first point of client's requirements (see section 2.2), whilst generating the games schedule.

2.3.6 Hardware

Table 4 illustrates the optimal and minimum hardware requirements needed to run the programme.

Item	Specification	
	Optimal	Minimum
Processor	1 GHz processor clock speed recommended; (dual processor system); Intel Pentium/Celeron family, or AMD K6/Athlon/Duron family, or compatible processor recommended	233 MHz minimum required (single processor system); Intel Pentium/Celeron family, or AMD K6/Athlon/Duron family, or compatible processor recommended
RAM	512 megabytes (MB) of RAM or higher recommended	(64 MB minimum supported; may limit performance)
Hard Disk	1.5 gigabytes (GB) of available hard disk space	1.5 gigabytes (GB) of available hard disk space
Monitor	Super VGA (800 x 600) or higher-resolution video adapter and monitor	VGA (800 x 600) or higher-resolution video adapter and monitor
Keyboard and Mouse	Keyboard and Microsoft Mouse or compatible pointing device	Keyboard and Microsoft Mouse or compatible pointing device

Table 4

2.3.7 Software

Table 5 illustrates the optimal and minimum software requirements needed to run the programme.

Item	Specification	
	Optimal	Minimum
OS	Windows 2000, or Windows XP	Microsoft Windows® 98, Windows 98 Second Edition,
software	Microsoft Office 2003	Microsoft Office 97

Table 5

3 Investigation (Task2)

3.1 Design Approaches (Methodologies) (Task2/a)

A methodology is a formalised approach or series of steps. It provides a framework, in which the design of the system can be modelled and improves efficiency of system development. The different methodologies or design approaches used in programme design are discussed in the following subsections.

3.1.1 Rapid Application Development (RAD) Design

RAD (**R**apid **A**pplication **D**evelopment) is a software development process which involves iterative development, the construction of prototypes, and the use of Computer-aided software engineering (CASE) tools.

Figure 1 (Visual Basic Programs Templates)

Products can be developed faster and of higher quality through:

- Gathering requirements using workshops or focus groups
- Prototyping and early, reiterative user testing of designs
- The re-use of software components
- A rigidly paced schedule that defers design improvements to the next product version
- Less formality in reviews and other team communication

RAD usually includes object-oriented programming methodology, which facilities and encourages software re-use. The most popular object-oriented programming languages, C++ and Java, are offered in visual programming packages often described as providing rapid application development. Visual Basic is a RAD, as it provides a fast way of producing applications (prototyping). [1]

3.1.2 Prototyping

Prototyping is the process of quickly putting together a working model (a prototype) in order to test various aspects of a design, illustrate ideas or features and gather early user feedback. Prototyping is often treated as an integral part of the system design process, where it is believed to reduce project risk and cost. [1]

3.1.3 Top-Down Design

Top-down approaches emphasise planning and a complete understanding of the system. It is inherent that no coding can begin until a sufficient level of detail has been reached in the design of at least some part of the system. This, however, delays testing of the ultimate functional units of a system until significant design is complete.

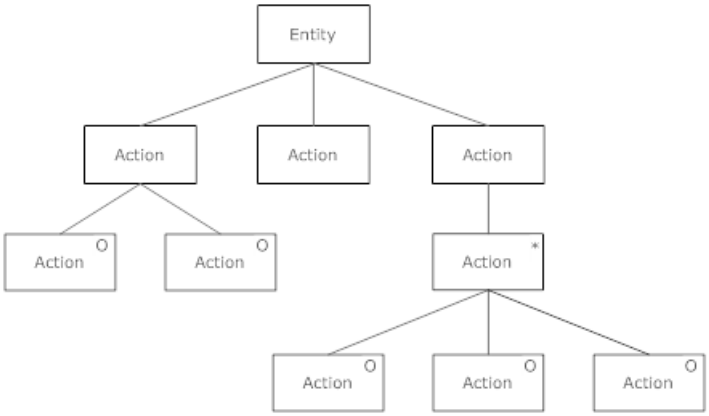


Figure 2 (Top-Down Design Model)

In this approach, the design begins by specifying complex pieces and then dividing them into successively smaller pieces. The smaller pieces or modules are then broken into smaller pieces until the remaining pieces can be coded and the programme is written. This process is called modularisation.

3.1.3.1 Steps in Modularisation

- Define the problem using the Defining Diagram.
- Group activities into subtasks or functions.
- Construct a hierarchy chart.
- Establish the mainline algorithm.
- Develop the algorithms for each successive module.
- Desk checks the solution algorithm.

3.1.3.2 Advantages of Top-Down Design

- Documentation is more likely to be accurate and up to date.
- The design can be meaningful to the client and the business analyst. This is because many of the technical details are in the lower levels of the modules.
- Testing can begin as soon as the first module is completed. This is done by using a test harness in which the parameters that are to be passed to the module are simulated to check the effectiveness of the module in handling data.
- Allocation to team members is easier because the programme is broken into many parts. this speeds up the development and implementation of the programme
- Because of modularisation, the code can be re-used in many different places of the programme and in different programmes.

Top-down programming is common in traditional procedural programming languages such as C and Pascal. [3]

3.1.4 Bottom up Design

Bottom-up Design approach emphasises coding and early testing, which can begin as soon as the first module has been specified. This approach, however, runs the risk that modules may be coded without having a clear idea of how they link to other parts of the system, and that such linking may not be as easy as first thought. Re-usability of code is one of the main benefits of the bottom-up approach. The bottom-up programming approach is common in object-oriented languages such as C++ or Java.

In this design approach the designer selects components already built and adds a control structure to link them together. Changes are made to adjust the low level modules to do what is wanted to complete the programme. This methodology is used when building similar programmes for different customers. Only a limited amount of customisation is needed to meet the slightly different needs of the different customers. This approach is also used in customising software packages such as accounting or spreadsheets where there are repeated business needs. Data Driven Design

The designer breaks down the design so that each module processes one of the main items of data rather one business function. It shares most of modularisation benefits as top-down design. However it is easier to use where the programme subroutines align

closely to the data structure, as in many data capture applications such as payroll systems, stock levels, etc. [1]

3.1.5 Event Driven Design

An event is an occurrence such as a key press, mouse move or click, which initiate an action by the programme. Event driven design is based on execution of the modules when a specific event occurs. This approach is used in interactive system in which actions depends on user wish. Visual Basic and Visual C++ are event programming tools.

3.2 Data Structures (Task2/b)

A data structure is how related pieces of information, including variables of different data types, are organised. The structures can be files, lists, arrays, records, trees or tables.

3.2.1 Data types

A data type specifies the sort of data the variable is expected to contain. The most common data types are explained below:

1. Characters: this data type stores letters a-z and A-Z and special characters such as & %$#(). A string is a collection of characters such as "hello".
2. Numeric: stores number which can be whole numbers or decimal
3. Boolean: binary data type which takes the value true (1) or false (0).
4. Date: stores a specific date which can be in many different formats such as 09/12/1977 and 09-Dec-1977.
5. Time: stores a specific time.

3.2.2 Files

A file is a collection of data that has a file name that the computer operating system knows about.

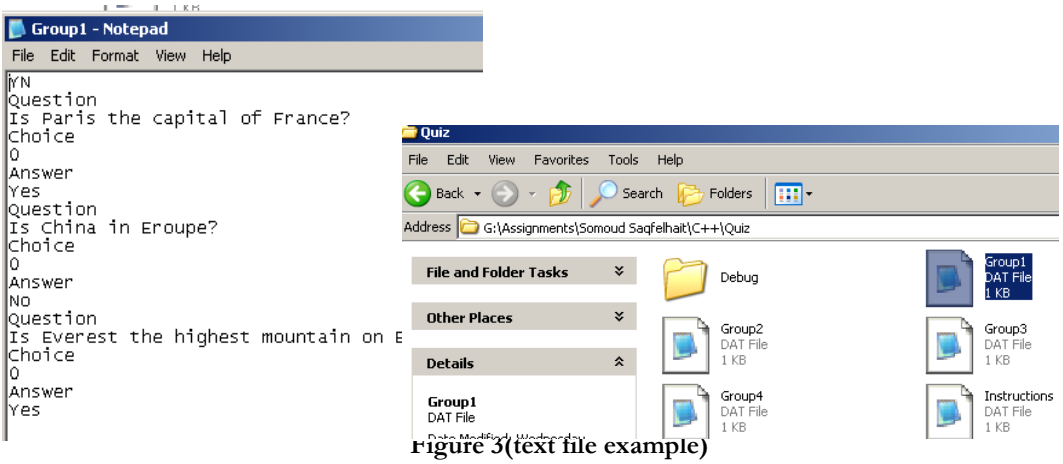


Figure 3(text file example)

3.2.3 Arrays

An array is a set of objects that are all the same size and data type, e.g. an array of names of students in a specific class

Visual Basic declaration: **`Dim classStudentNames(18) As String`**

This declaration line tells the computer that a block of memory should be reserved to store names of eighteen students.

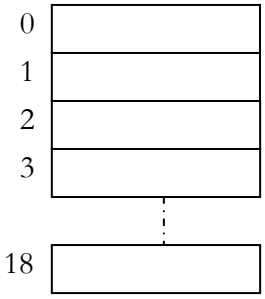


Figure 4(classStudentNames Array)

To assign a value, i.e. a name to place in the array

`classStudentNames(0)= "Arnold"`

`classStudentNames(1)= "Albert"`

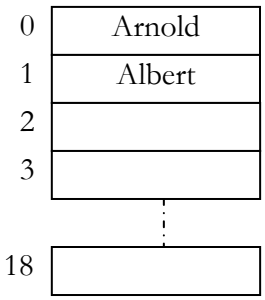


Figure 5(classStudentNames Array)

3.2.4 Records

A record is like one dimensional array in that it is comprised of a series of related data items. Records differ from arrays in that, whereas all the elements in an array have the same type, the successive data items in a record may differ in type. Example of records is student records. Each student would have a number of attributes such as name, ID number, address, etc. [2]

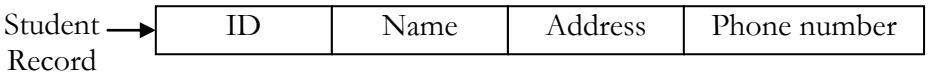


Figure 6(record)

3.2.5 Lists

Provide a flexible way of handling data items in order. Changes to the order can be achieved with minimal data movement and little loss of storage place. [1]

Lists can be used to store a list of records. The items in a list can be sorted or unsorted. There are special types of lists such as linked-lists, queues and stacks. Two examples of lists implemented in GUIs (Graphical User Interface) are List Boxes and Combo Boxes

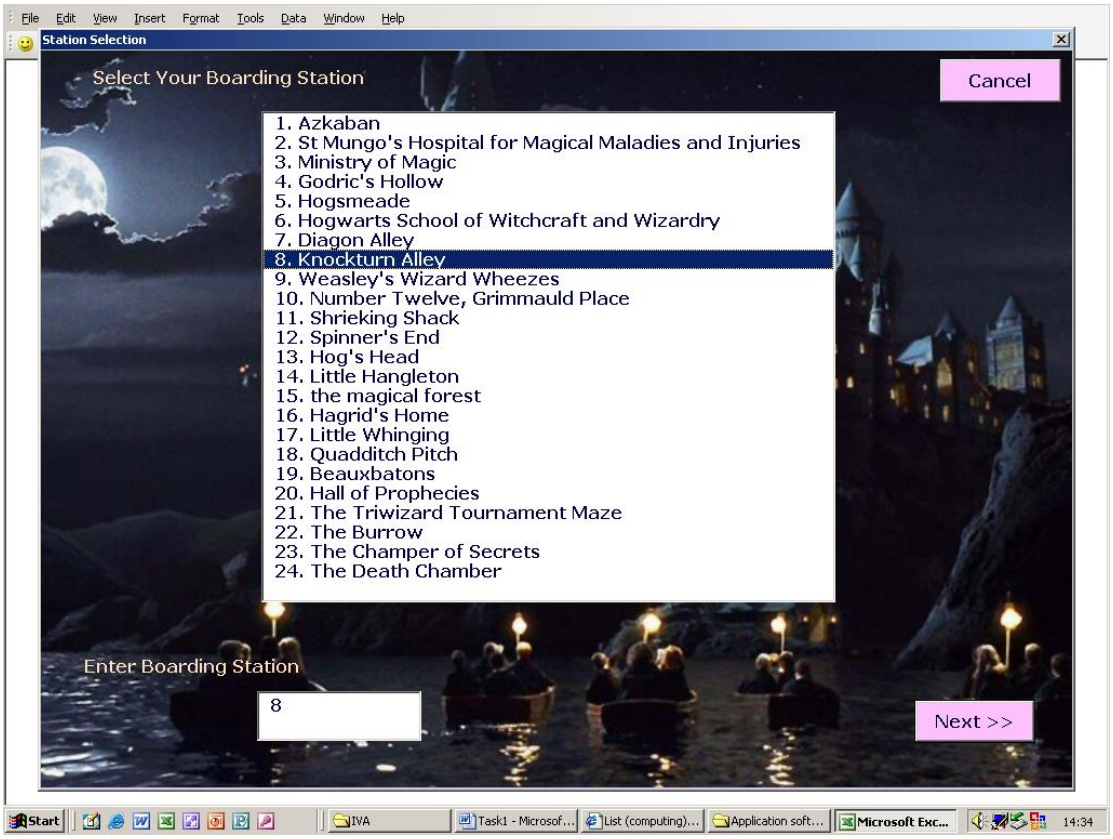


Figure 7(list box example)

Other examples of listed are linked lists, Queues and stacks.

3.2.6 Trees

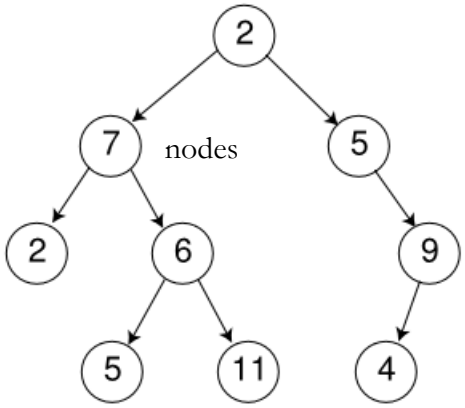


Figure 8 (binary tree)

Trees are hierarchical data structures rather like the familiar family tree. They are constructed using rule of precedence for the data items, e.g., using alphabetic or numerical sequence. The elements of a tree are called nodes and each element consists of a datum and at least two pointers. [3]

Trees are usually used to manipulate hierarchical data, make information easy to search and manipulate sorted lists of data

3.2.7 Tables

A table is a set of rows and columns containing a variable number of entries. This similar to a record except that records are usually read sequentially [1]

Enquiry Ref.	Date	Holiday Reference Number	Boarding name	Destination name	Number of Passengers	Total Charge
1	2/22/2007	1119	St Mungo's Hospital for Magical Maladies and Injuries	Quadditch Pitch	3	£4.75
2	2/22/2007	1101	Beauxbatons	the magical forest	2	£10.50
3	2/22/2007	1104	Shrieking Shack	The Champer of Secrets	7	£14.81
5	2/22/2007	1114	Knockturn Alley	The Champer of Secrets	3	£12.00
6	2/22/2007	1117	Hogwarts School of Witchcraft and Wizardry	The Death Chamber	5	£14.75

Figure 9(Table)

4 Design (Task3)

Top-down design approach will be used to design the required programme

4.1 Programme's State Diagram(Task3/a/c)

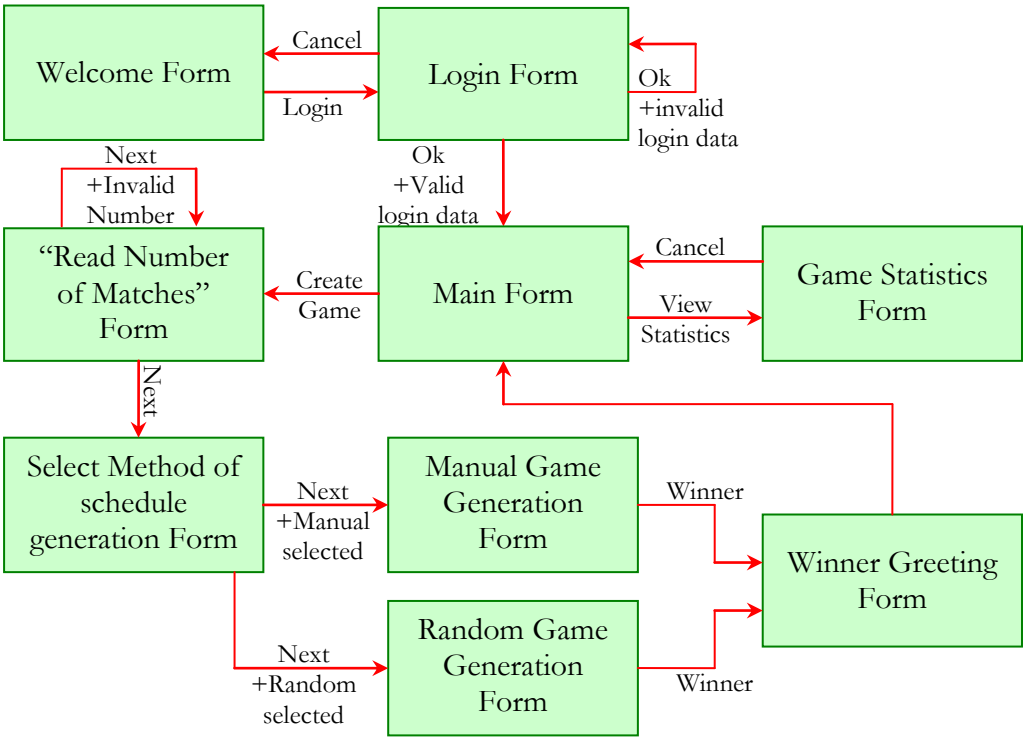


Figure 10

4.2 Structure Diagrams and Pseudo Code (Task3/a/c)

The programme will be divided into functions

4.2.1 Tournament Organising System

The main processes involved are:

- Input the number of matches.
- Generating the games schedule.
- Input the results.
- Calculate each house score
- Decide who the winner is.

Although the processes are listed as a sequence, they are event driven processes; that means each of those processes will be initiated as a result of user input (clicking a command button or pressing a key).

4.2.1.1 Structure Diagram

The above processes will form the upper layer of the programme structure diagram as shown in figure 10.

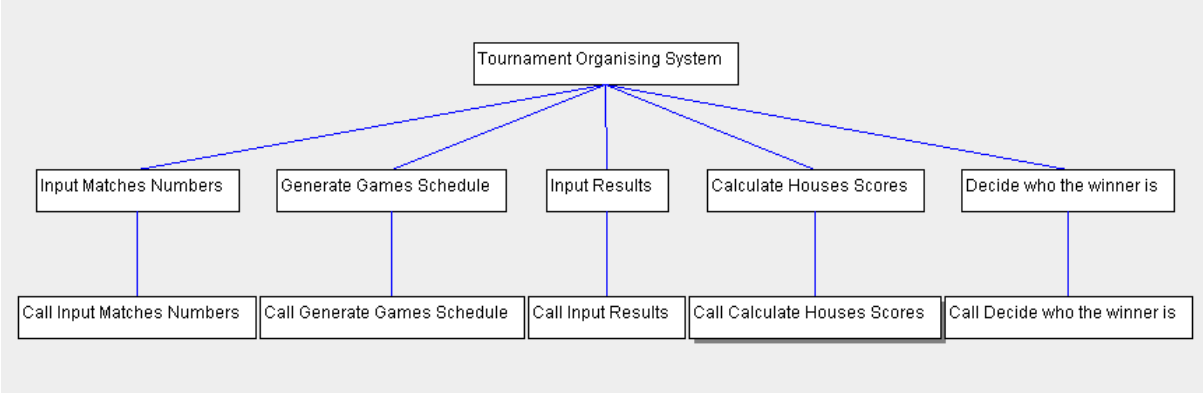


Figure 11(main structure diagram)

4.2.1.2 Pseudo Code

```
Begin
        Call Input Matches Numbers
        Call Generate Games Schedule
        Call Input Results
        Call Calculate Houses Scores
        Call Decide who the winner is
End
```

Figure 12(main pseudo code)

4.2.2 Input Matches Numbers

Taking into account the rules of the game listed in the client requirements, section (2.2), and the fact that the number of competing students from each house is eighteen; the maximum number of matches for each game is eighteen, the minimum is one match. So the user will be allowed to enter a number between one and eighteen for each game. If the user where to choose games' match numbers so that the sum of all of them is an even number, there is a possible draw between the two houses. I had two solutions in mind for this problem. The first is to force the sum of the games' match numbers to be an odd number. The second is to wait until the results are entered and if a draw occurs a deciding match is held. I will chose the first solution because if the user enters the maximum number for each game, playing a deciding game will be against the roles (see section 2.2 /point 1).

4.2.2.1 Structure Diagram

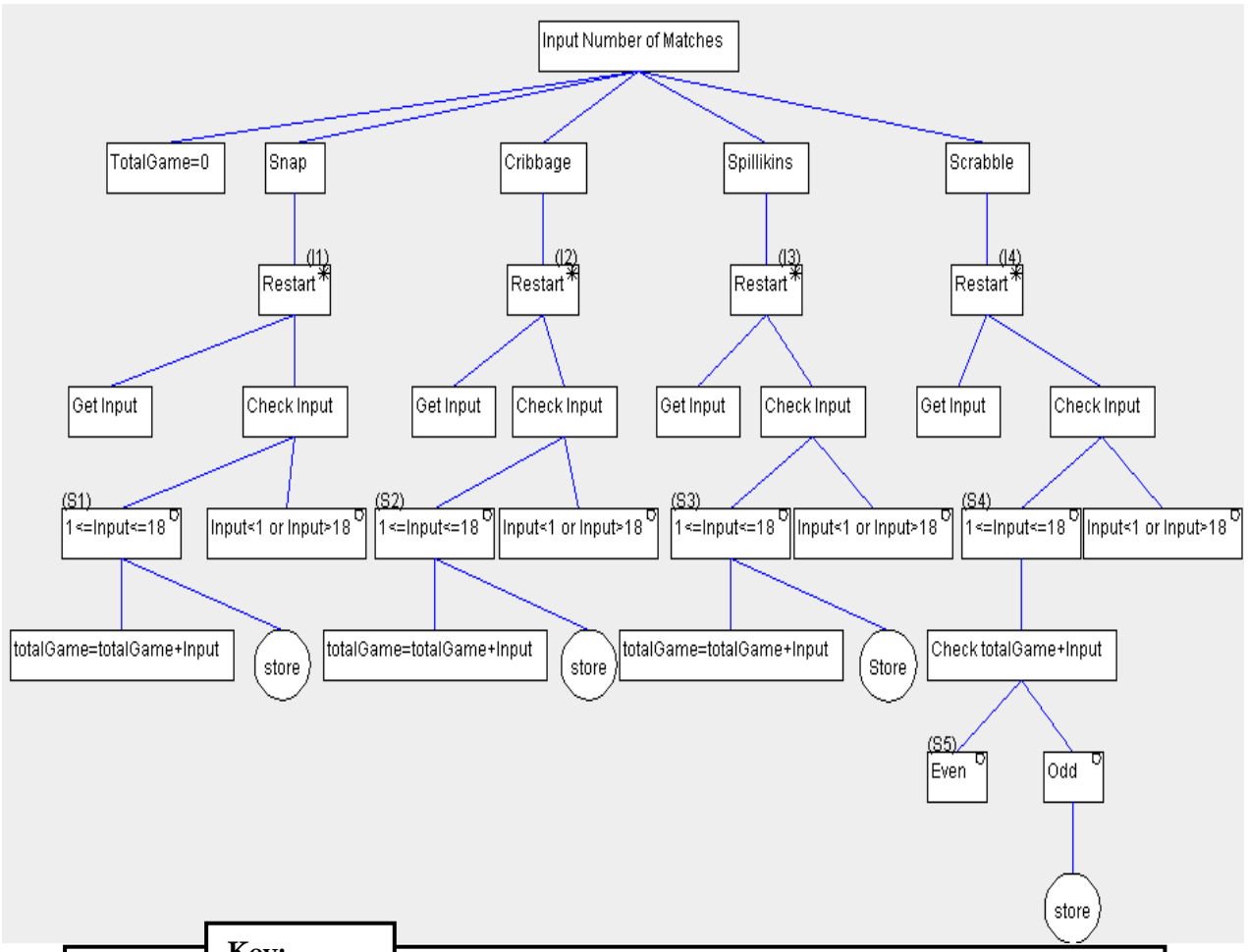


Figure 13(Input Number of Matches Structure Diagram)

4.2.2.2 Pseudo Code

```
Begin
 TotalGame=0
 // read snap number of matches
 While Invalid Input
 Begin
  Get Input
  If 18>=Input>=1
   totalGame=totalGame+Input
    Store input in SnapNo.
 End
 // read cribbage number of matches
 While Invalid Input
 Begin
  Get Input
   If 18>=Input>=1
     totalGame=totalGame+Input
      Store input in CribbageNo.
 End
 // read snap number of matches
 While Invalid Input
 Begin
  Get Input
    If 18>=Input>=1
      totalGame=totalGame+Input
      Store input in SpillikinsNo.
 End
// read scrabble number of matches
While Invalid Input
Begin
  Get Input
   If 18>=Input>=1
     If totalGame+Input is odd
          Store input in ScrabbleNo.
 End
End
```

Figure 14(Input Number of Matches Pseudo Code)

4.2.3 Generate Games Schedule

4.2.3.1 Structure Diagram

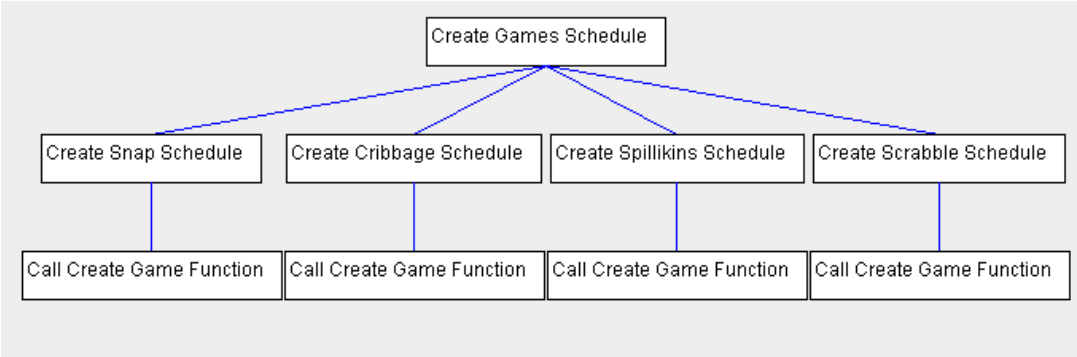


Figure 15(Create Games Schedule Structure Diagram)

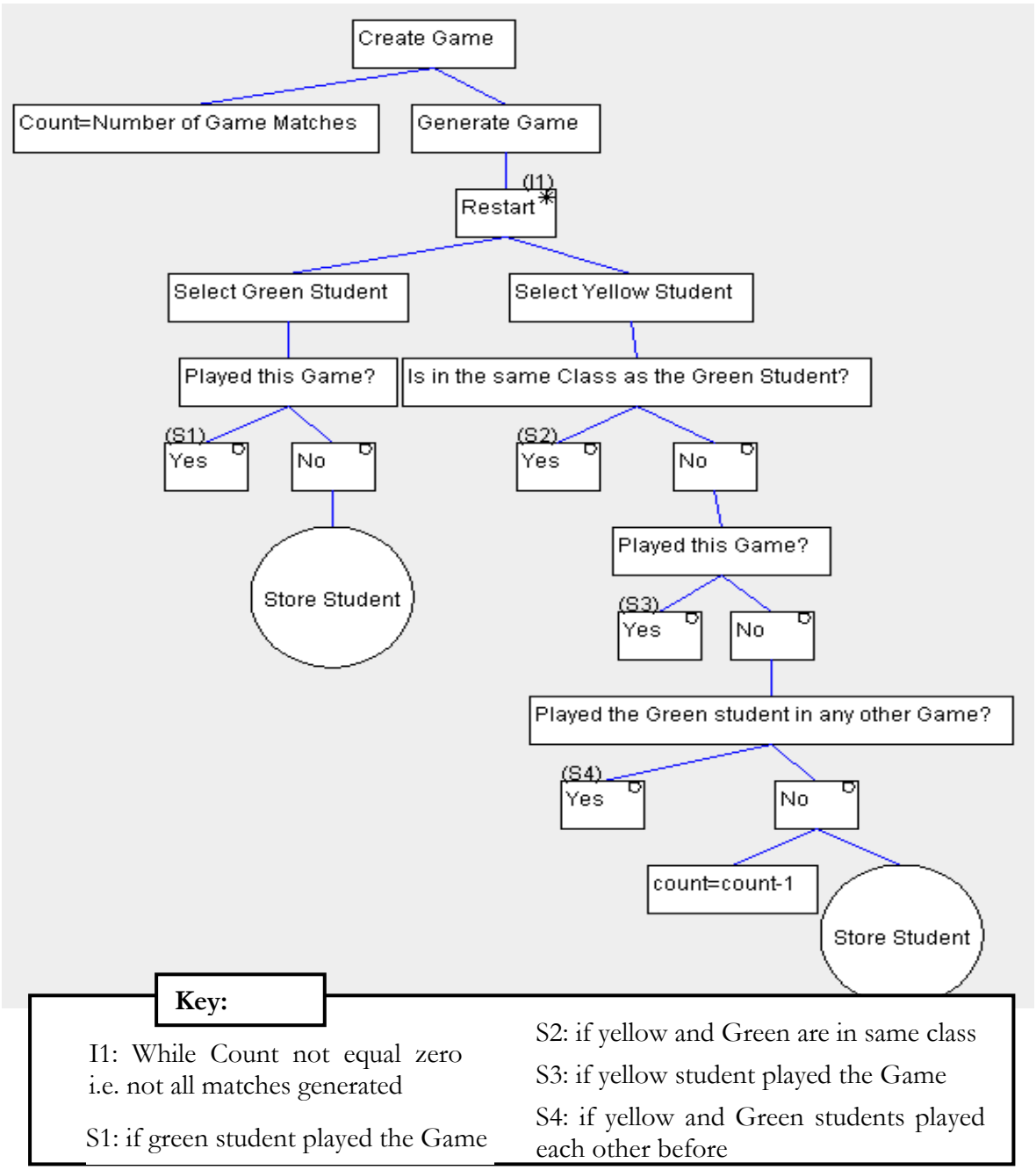


Figure 16(Create Game Function Structure Diagram)

4.2.3.2 Pseudo Code

```
Begin
        Call Create Game Function for Snap
        Call Create Game Function for Cribbage
        Call Create Game Function for Spillikins
        Call Create Game Function for Scrabble
End
```

Figure 17(Create Games Schedule Pseudo Code)

```
Inputs: Game Name, Number of Matches
 Students names, classes and houses
Begin
        Count=Number of Game Matches
        While count not equal zero
        Begin
                Select a Green student randomly
                If student did not play this game before
                        (* Operation Store Student *)
                Select a Yellow student randomly
                If YellowClass not equal GreenClass
                If student did not play this game before
                If Green and Yellow did not compete in any other game
                        (* Operation Store Student *)
                        count =count-1
        End
End
```

Figure 18(Create Game Function Pseudo Code)

4.2.4 Read Results

4.2.4.1 Structure Diagram

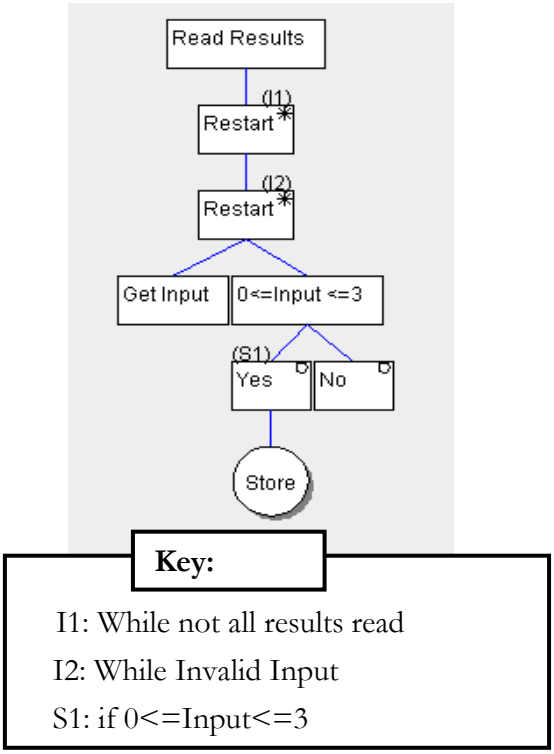


Figure 19(Read Results Structure Diagram)

4.2.4.2 Pseudo Code

```
Begin
  While not all results read
        Begin
        While Invalid Input
              Begin
                    Get Input
                    If 0<=Input<=3 then
                          Store Result
              End
        End
End
```

Figure 20(Read Results Pseudo Code)

4.2.5 Calculate Houses Scores

4.2.5.1 Structure Diagram

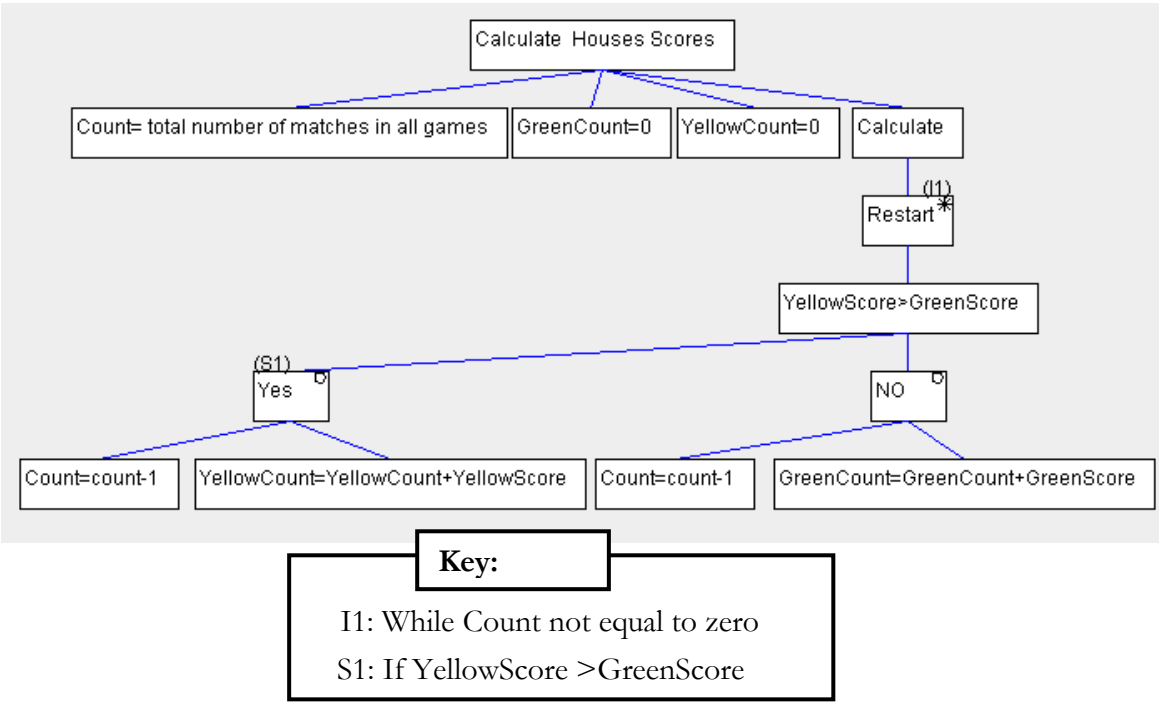


Figure 21(Calculate Houses Scores Structure Diagram)

4.2.5.2 Pseudo Code

```
Begin
        Count= total number of matches in all games
        GreenCount =0
        YellowCount =0
        While Count not equal to zero
        Begin
                If YellowScore >GreenScore
                        Count=count-1
                        YellowCount=YellowCount+YellowScore
                Else
                        Count=count-1
                        GreenCount=GreenCount+GreenScore
        End
End
```

Figure 22(Calculate Houses Scores Pseudo Code)

4.2.6 Decide On the Winner

4.2.6.1 Structure Diagram

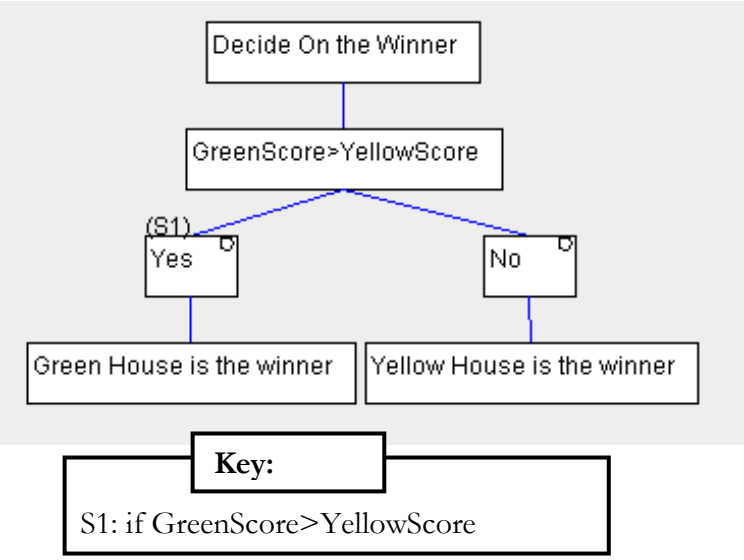


Figure 23(Decide On the Winner Function Structure Diagram)

4.2.6.2 Pseudo Code

```
Inputs: GreenScore, YellowScore
Begin
        If GreenScore>YellowScore
                Green House is the winner
        Else
                Yellow House is the winner
End
```

Figure 24(Decide On the Winner Function Pseudo Code)

4.3 Data structures (Task3 /c)

Two arrays will be used to store students of each house. A two dimensional array will be used to store the opponents in each game.

4.4 Data storage (Task3 /c)

Access file will be used to store the names of students and their classes and houses

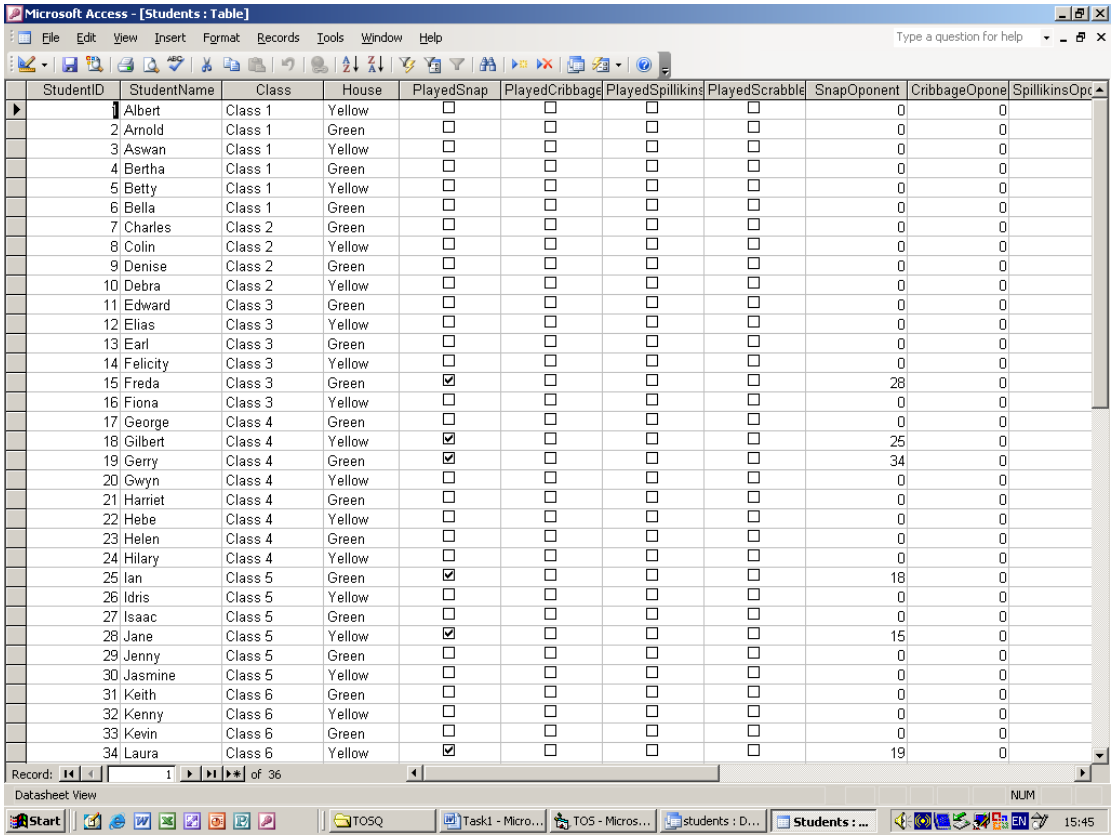


Figure 25(Access database)

The score of each house will be stored in two integer variables. The names of the students in each house will be read from the access file and stored in two string arrays. The competition student pairs in all games will be stored in a string array once they are generated in the method chosen by the user (random or manual).

4.5 Inputs (Task3 /c)

- Number of Snap matches.
- Number of Cribbage matches.
- Number of Spillikins matches.
- Number of Scrabble matches.
- Results.

4.6 Outputs (Task3 /c)

- Array of Games matches.
- Green House Score.
- Yellow House Score.
- The winner.

4.7 Processes (Task3 /c)

- Select students to play games.
- Compute the green house score.
- Compute the yellow house score.

4.8 User Interface (Task3 /c)

4.8.1 Login Form

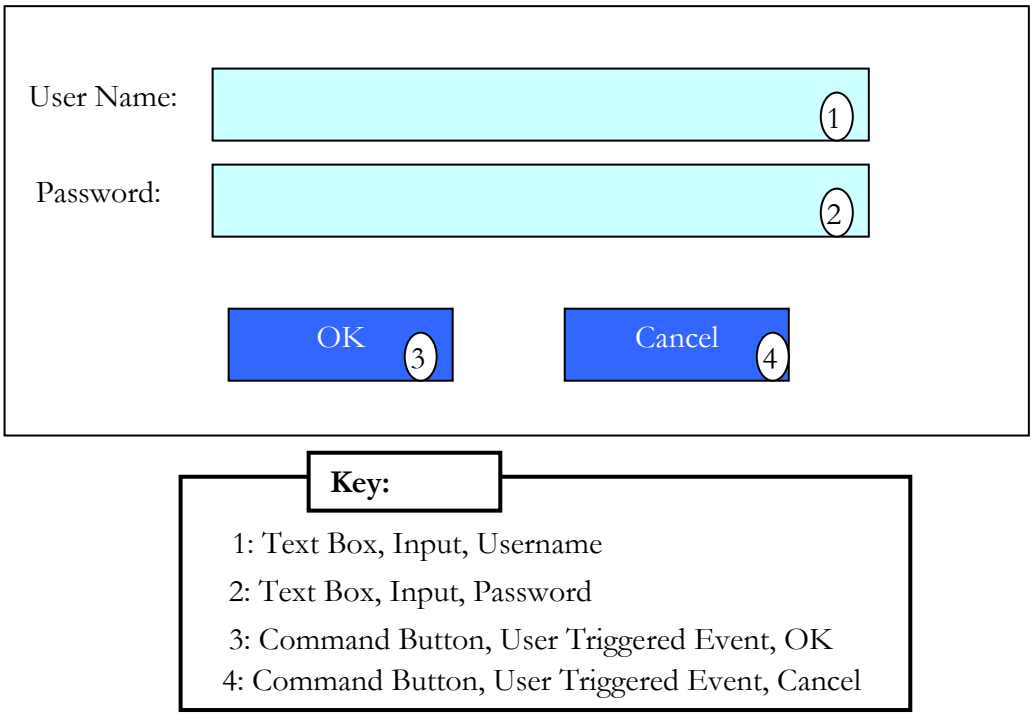
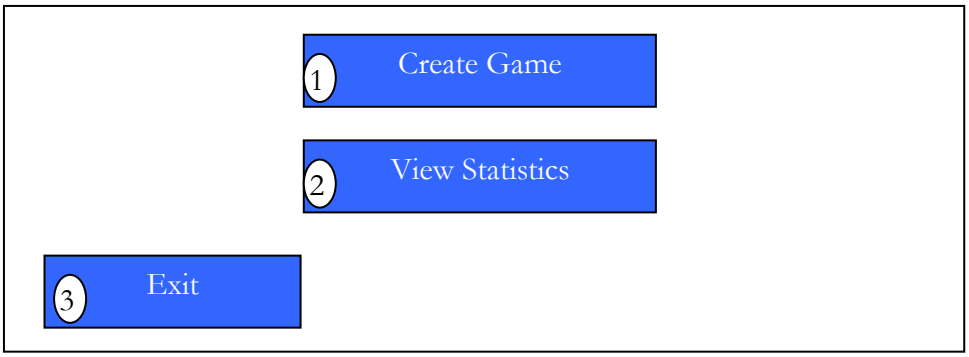


Figure 26(Login Form Layout)

This form will be used to authorise access to the rest of the programme. On entry of a username "1" and password "2" text boxes, and when the OK"3" command button is clicked. The username and password entered will be checked against the stored username and password, if they match the user will gain access to the programme. If not, they will be asked to enter the correct combination of username and password. The Cancel "4" command button will allow the user to exit the login screen without logging in.

4.8.2 Main Form



Key:

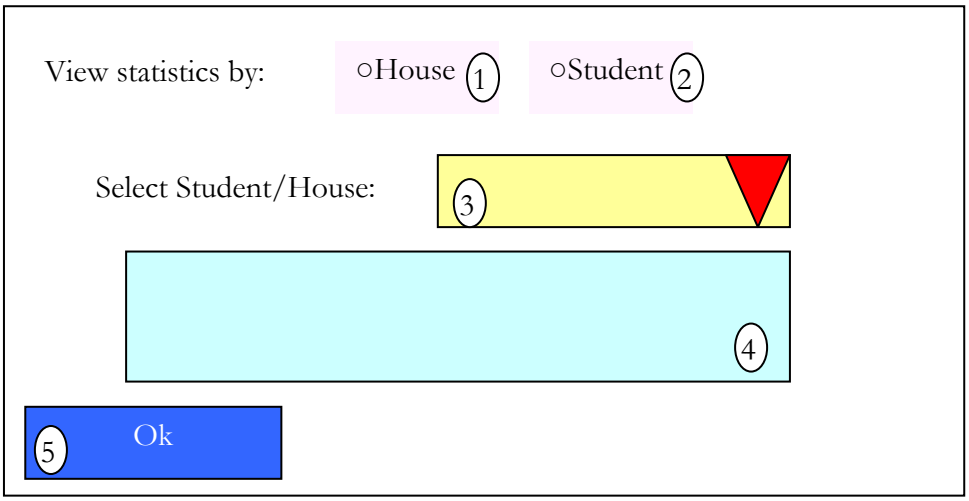
1: command button, loads Read Number of Matches Form

2: command button, loads Statistics Form

3: command button, exit the programme

Figure 27(Main Form Layout)

4.8.3 Statistics Form



Key:

1: Option button

2: Option button

3: Combo Box, Input, Value depend on the option button selected

4: Label, display requested information

5: Command Button, return to the main form

Figure 28(Statistics Form Layout)

4.8.4 Read Number of Matches Form

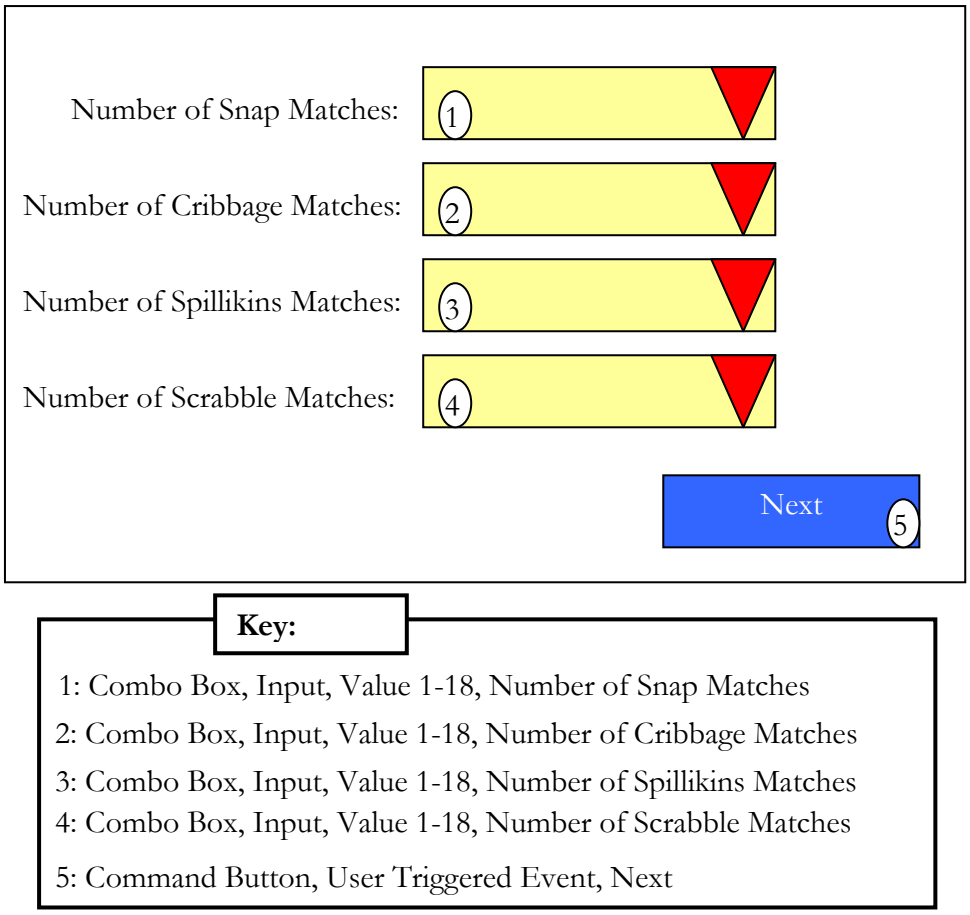


Figure 29(Select Number of Matches Form Layout)

This form will allow the user to enter the number of matches for each game. The number of each match should be in the range 1-18. Validation will be used to restrict the input to this range. The validation will take place when the "Next" button is clicked. If the user entered valid numbers, the game schedule method of generation selection form is loaded.

4.8.5 Game Generation Method Selection Form

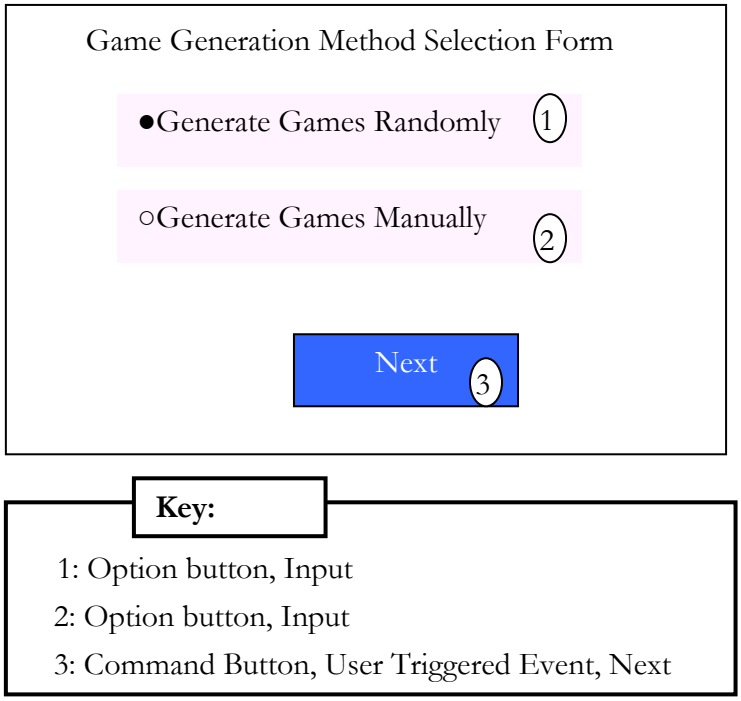
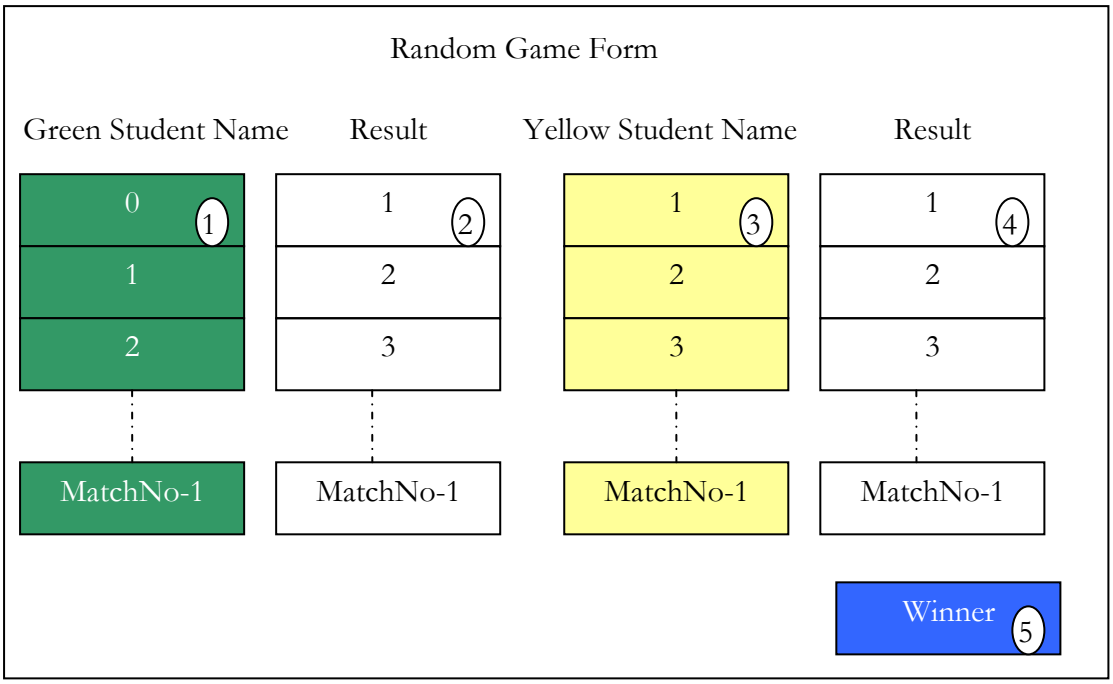


Figure 30(Game Generation Method Selection Form Layout)

This form will be used to allow the user to decide the way they want the game schedule to be generated; manual or random. When the "Next" is clicked the form corresponding to the user selection will be showed.

4.8.6 Random Game Schedule Form



Key:

1: Control array of labels, display green house opponent name
2: Control array of text boxes, used to input green student score, value 0-3
3: Control array of labels, display yellow house opponent name
4: Control array of text boxes, used to input yellow student score, value 0-3
5: Command Button, User Triggered Event, Winner

Figure 31(Random Game Schedule Form Layout)

This form will be loaded if the user chooses to enter the names of the games contenders (Random). The labels will be used to display the opponents in each game. The result text boxes will be used to allow the user to enter the results of the games. When entering the result for the green or yellow student the programme will calculate automatically the score of the opponent. When the user click "Winner" button, the programme will check if all results are entered. If not it will highlight the text boxes that need to be filled. Once all text boxes are filled and the "Winner" button is clicked, the programme calculates the score of each house and compares scores to decide the winner. The number of boxes in 1, 2, 3 and 4 will depend on the number of matches for each game. There will be a form for each game.

4.8.7 Manual Game Schedule Form

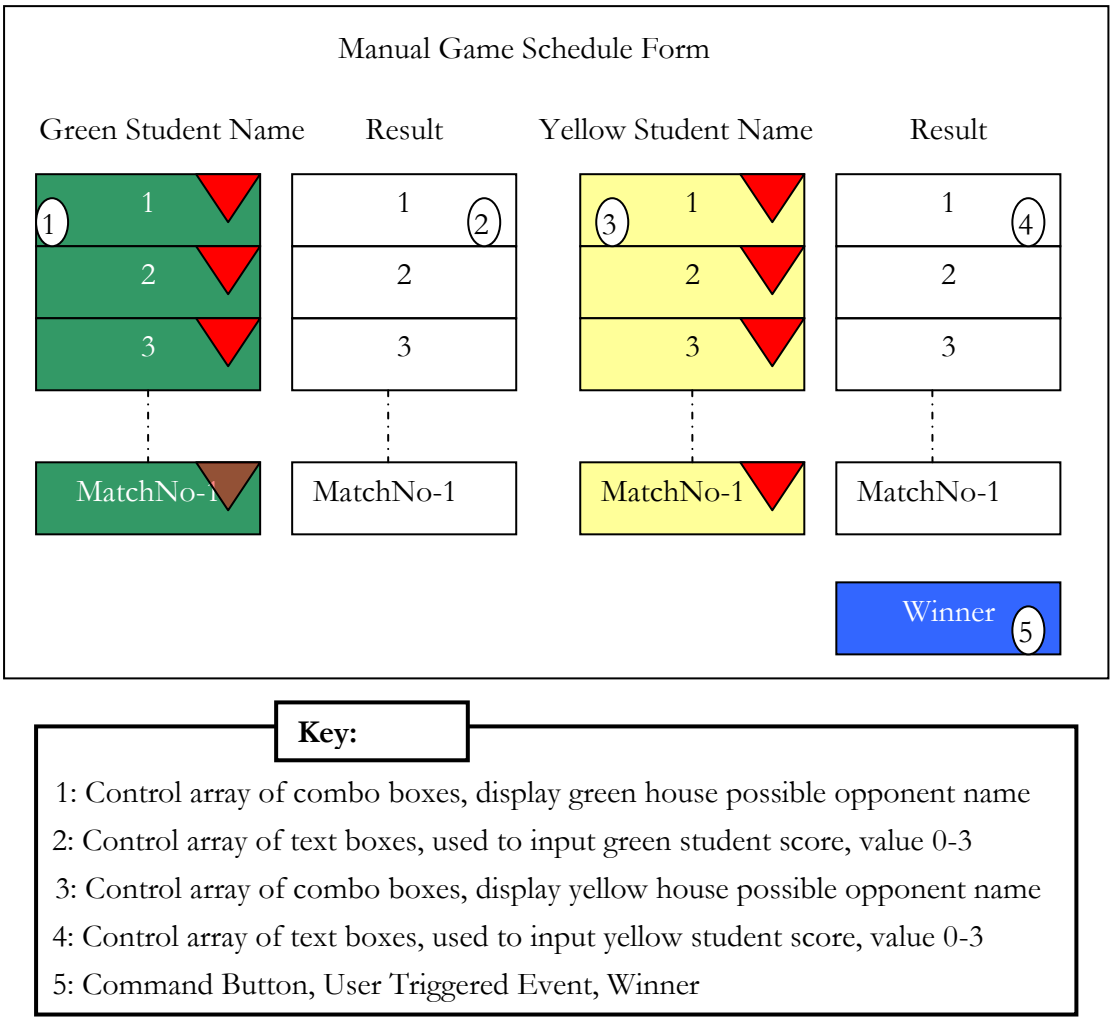


Figure 32(Manual Game Schedule Form Layout)

This form will be loaded if the user chooses to enter the names of the games contenders manually. The combo boxes will be updated automatically to allow the user to choose the student allowed playing in that position. In other words a list of all possible players in that position will be displayed in the combo box list. The result text boxes will be used to allow the user to enter the results of the games. When entering the result for the green or yellow student the programme will calculate automatically the score of the opponent. When the user click "Winner" button, the programme will check if all results are entered. If not it will highlight the text boxes that need to be filled. Once all text boxes are filled and the "Winner" button is clicked, the programme calculates the score of each house and compares scores to decide the winner. The number of boxes in 1, 2, 3 and 4 will depend on the number of matches for each game. There will be a form for each game.

4.9 Data Dictionary(Task3 /c)

4.9.1 Data Dictionary-Functions

Function Name	Return Type	Arguments/Parameters		Description
		Name	Data type	
playedGame	Boolean	studentName	string	This function will decide if the parameter studentName from house has played this Game. if played it will return true otherwise false
		house	string	
		thisGame	String	
playedStudent	Boolean	studentName	string	This function will decide if studentName has played opponent in thisGame. If they played each other in thisGame it will return true otherwise it will return false
		opponent	string	
		thisGame	string	
winner	String	greenScore	integer	This function will decide who on the winner passed on the parameters. If greenScore is greater than yellowScore it will return "Green" otherwise it will return "Yellow"
		yellowScore	integer	

Table 6

4.9.2 Data Dictionary –Procedures

Procedure Name	Related Event	Arguments/ parameters		Description
		Name	Data type	
createGameRand	Click next command button in method selection form and random option button was selected (figure 27)			This procedure loads generate the games schedule function based on user input (random).
createGameMan	Click next command button in method selection form and manual option button was selected (figure 27)			This procedure loads generate the games schedule function based on user input (manual).
calculateResult	Click winner command button in game schedule forms (figures 28 and 29)			This procedure decides on the winner and loads the greeting form with the winning house colours
login		username	string	This procedure prevent unauthorised access to the programme
		password	string	

Table 7

4.9.3 Data Dictionary –Variables

Identifier Name	Data type	Description
GreenStudents	Array of strings	An array of strings used to store green house student
YellowStudents	Array of strings	An array of strings used to store yellow house student.
GreenScore	integer	An integer variable used to collect green house score.
YellowScore	integer	An integer variable used to collect yellow house score.
Winner	string	A string variable used to store the name of the winning house.
Games	Array of strings	An array of strings used to store names of opponents in each game.

Table 8

4.10 Test plan (Task3 /b)

Testing is the process of checking that all the functions of the programme work as they should and give correct results. Besides, testing ensures that the programme is robust and can withstand being used incorrectly without crashing. The two types of testing I will be using are introduced in the following sub-sections.

4.10.1 White Box Testing

This method involves inspecting the code of the programme so that errors can be identified. I will use trace tables to trace the values of the main variables in the programme using the debugging tools provided with Microsoft Visual Studio such as "step into", "step over", "run to cursor" ,break point, watch value at run time … etc.

4.10.2 Black Box Testing

In black box testing, we concentrate on what the programme is supposed to do and we don't need to look at the code.

I will select test cases, which are specific inputs that I will tryout, and procedures that I will be following to test the programme.

I will use equivalence partitioning to select a set of test cases. Equivalence partitioning is the process of reducing the infinite number of test cases into much smaller, but effective set. This ensures a proper selection of test cases and ensures accurate test results.

I start testing using test-to-pass approach; that is choose a straight forward test set and see if the programme runs as expected. I will then use test-to fail approach to find programme bugs; this approach involves using unexpected inputs. The black box testing is listed in the following sub-sections

4.10.2.1 Login Form Test Plan

To test each control, the correct data will be entered in all controls except the one that we are testing. Table 9 is the test plan for login form, the design of the form is shown in figure 26/page 23. The numbers in the Test field of the table indicate the label number of the control in the layout diagram.

Test no.	Test	Test data	Expected Result
1	1 (username text box)	blank	When 3 "OK" is clicked a message will appear to ask the user to enter the correct username
2	1 (username text box)	Correct username	When 3 "OK" is clicked the form in figure 27/page 24 (main form) will appear
3	1 (username text box)	incorrect username	When 3 "OK" is clicked a message will appear to ask the user to enter the correct username
4	2 (password text box)	blank	When 3 "OK" is clicked a message will appear to ask the user to enter the correct password
5	2 (password text box)	Correct password	When 3 "OK" is clicked the form in figure 27/page 24 (main form) will appear
6	2 (password text box)	incorrect password	When 3 "OK" is clicked a message will appear to ask the user to enter the correct password
7	4 (Cancel command button)	click	The process will be cancelled and the programme exited

Table 9

4.10.2.2 Read Number of Matches Form Test Plan

To test each control, the correct data will be entered in all controls except the one that we are testing. . Table 10 is the test plan for Read number of matches form, the design of the form is shown in figure 29/page 25. The numbers in the Test field of the table indicate the label number of the control in the layout diagram.

Test no.	Test	Test data	Expected Result
1	Snap, Cribbage, Spillikins and scrabble match numbers 1,2,3,4	Unchanged i.e. blank	When 5 "Next" is clicked a message will appear to ask the user to enter valid data 1-18
2	1,2,3,4 (combo boxes)	0	When 5 "Next" is clicked a message will appear to ask the user to enter valid data 1-18
3	1,2,3,4 (combo boxes)	a	When 5 "Next" is clicked a message will appear to ask the user to enter valid data 1-18
4	1,2,3,4 (combo boxes)	3	When 3 "Next" is clicked the select generation method form (figure30/page26) will be loaded

Table 10

4.10.2.3 Game Generation Method Selection Form Test Plan

To test each control, the correct data will be entered in all controls except the one that we are testing. Table 11 is the test plan for game generation method selection form, the design of the form is shown in figure 30/page 26.

Test no.	Test	Test data	Expected Result
1	1 (option buttons) random	Selected/ 2 unselected	When 3 "Next" is clicked the random game form (figure31/page27) will appear with a randomly generated game schedule
2	2 (option buttons) manual	Selected/1 unselected	When 3 "Next" is clicked the manual generation form (figure32/page28) will appear with combo boxes to allow the user to create games schedule manually

Table 11

4.10.2.4 Random Game Schedule Form Test Plan

Table 12 is the test plan for random game schedule form, the design of the form is shown in figure 31/page 27.

Test no.	Test	Test data	Expected Result
1	2 (text boxes control array) green results	Blank	When 5 "Winner" is clicked the programme will highlight all the blank text boxes and ask the user to enter valid results
2	2 (text boxes control array)	a	When the text box lose focus a message box will appear and ask the user to enter valid result 0-3
3	2 (text boxes control array)	3	When the text box loses focus the corresponding text box "4" will be updated with its result automatically.
4	2 (text boxes control array)	4	When the text box lose focus a message box will appear and ask the user to enter valid result 0-3
5	4 (text boxes control array) yellow results	blank	When 5 "Winner" is clicked the programme will highlight all the blank text boxes and ask the user to enter valid results
6	4 (text boxes control array)	a	When the text box lose focus a message box will appear and ask the user to enter valid result 0-3
7	4 (text boxes control array)	3	When the text box loses focus the corresponding text box "1" will be updated with its result automatically.
8	4 (text boxes control array)	4	When the text box lose focus a message box will appear and ask the user to enter valid result 0-3
9	4 (command button)	When 5 "Winner" is clicked and all results are filled the programme will calculate the score and show greeting form with the winning house colour	

Table 12

4.10.2.5 Manual Game Schedule Form Test Plan

To test each control, the correct data will be entered in all controls except the one that we are testing. Table 13 is the test plan for manual game schedule form, the design of the form is shown in figure 32/page 28. The numbers in the Test field of the table indicate the label number of the control in the layout diagram.

Test no.	Test	Test data	Expected Result
1	1 (combo boxes control array) green students	blank	When 5 "Winner" is clicked a message will appear to ask the user to fill in all opponents in the game
2	1 (combo boxes control array)	A student chosen	When 5 "Winner" is clicked and all results are filled the programme will calculate the score and show greeting form with the winning house colour
3	3 (combo boxes control array) yellow students	blank	When 5 "Winner" is clicked a message will appear to ask the user to fill in all opponents in the game
4	3 (combo boxes control array)	A student chosen	When 5 "Winner" is clicked the programme will highlight all the blank text boxes and ask the user to enter valid results
5	2 (text boxes control array) green results	Blank	When 5 "Winner" is clicked the programme will highlight all the blank text boxes and ask the user to enter valid results
6	2 (text boxes control array)	a	When the text box lose focus a message box will appear and ask the user to enter valid result 0-3
7	2 (text boxes control array)	3	When the text box loses focus the corresponding text box "4" will be updated with its result automatically.
8	2 (text boxes control array)	4	When the text box lose focus a message box will appear and ask the user to enter valid result 0-3
9	4 (text boxes control array) yellow results	blank	When 5 "Winner" is clicked the programme will highlight all the blank text boxes and ask the user to enter valid results
10	4 (text boxes control array)	b	When the text box lose focus a message box will appear and ask the user to enter valid result 0-3
11	4 (text boxes control array)	3	When the text box loses focus the corresponding text box "1" will be updated with its result automatically.
12	4 (text boxes control array)	4	When the text box lose focus a message box will appear and ask the user to enter valid result 0-3

Table 13

4.10.2.6 Main Form Test Plan

Table 14 is the test plan for manual game schedule form, the design of the form is shown in figure 27/page 24. The numbers in the Test field of the table indicate the label number of the control in the layout diagram.

Test no.	Test	Test data	Expected Result
1	1 (Command Button) create game	-	When clicked will load "Read Number of Matches form"(figure29/page25)
2	2(Command Button) view statistics	-	When clicked will load "Game Statistics form"(figure28/page24)
3	3(Command Button) exit	-	A message box will be displayed to ask the user to confirm if he/she want to exit the programme

Table 14

4.10.2.7 Statistics Form Test Plan

Table 14 is the test plan for manual game schedule form, the design of the form is shown in figure 28/page 24. The numbers in the Test field of the table indicate the label number of the control in the layout diagram.

Test no.	Test	Test data	Expected Result
1	1 (house option button)	-	3-"student house combo box" should show the house names
2	2(student option button)	-	3-"student house combo box" should show the student names
3	3(combo box student/house)	-	If 1(house) is selected and a house name is chosen from this combo box statistics about that house should show in 4-list box
4	3(combo box student/house)	-	If 2(student) is selected and a student name is chosen from this combo box statistics about that student should show in 4-list box
5	5(Ok command button)	-	The form will be unloaded and the main form will be shown

Table 15

4.10.2.8 Algorithm testing

The method adopted for generating the game schedule will be implemented as a prototype first to make sure that it is valid. The results will be tested to check if they satisfy the client specifications.

5 Implementation (Task4)

The diagram in figure 33 shows the State diagram of the Tournament Organising system. The state diagram is used to describe the behaviour of the system. It describes all of the possible states of each form as events occur. And it is a summary of the transitions between the forms within the TOS. The arrows represent events, in this case clicking command buttons.

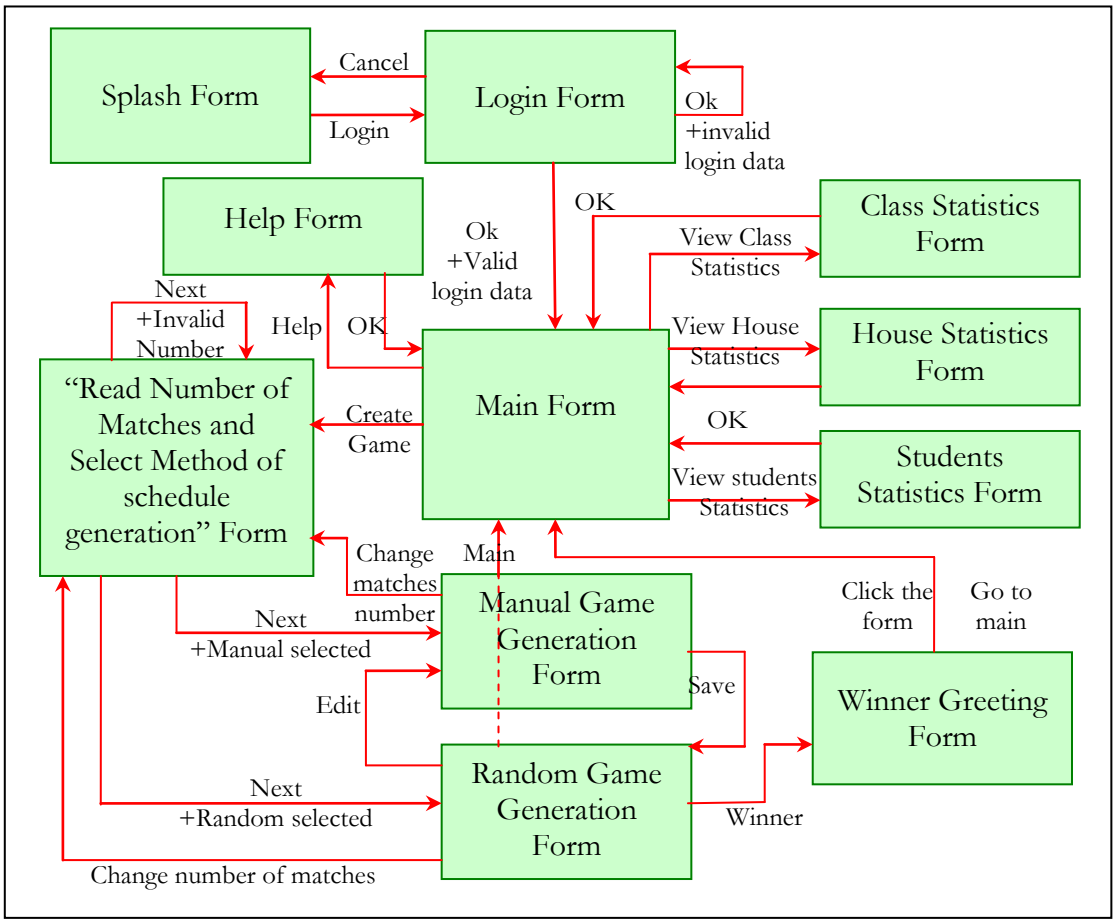


Figure 33 (Programme's state diagram)

I have used a module file to define public variables, procedures and functions used by all the forms in the programme

5.1 Module Code(mdlTOS.bas)(task4/a)

mdlTOS - 1

```
'********************************************************************************
'************************* Tournament Organising System************************
'*********************************modTOS Code**********************************
'*******************************Programer: S. Saqfelhait************************
'*********************************Date:07/04/2007*****************************
'this module contains functions and procedures that can be called from any form
'within the project
'********************************************************************************

'enforce declaration of all variables used within the module
Option Explicit

'store customised msgbox response
Public msgResponse As String

'a global variable used to store the winning house
Public gstrWinner As String

'a global variable to store the way in which the user choose to generate the games
'true if games to be generated randomly otherwise false
Public gboolRandom As Boolean

'a global variable to indicate that the game is being edited
Public gboolEdit As Boolean

'a global variable used to store the number of snap matches
Public gintSnapMatch As Integer

'a global variable used to store the number of cribbage matches
Public gintCribbageMatch As Integer

'a global variable used to store the number of spillikins matches
Public gintSpillikinsMatch As Integer

'a global variable used to store the number of scrabble matches
Public gintScrabbleMatch As Integer

'define a variable to connect to Access Database
Public gadoConnection As New ADODB.Connection

'define a variable to hold SQL commands
Public gadoCommand As New ADODB.Command

'define a variable to execute SQL commands and to access Access records
Public gadoRecordSet As ADODB.Recordset

'define a variable to hold the connection path
Public gstrConnection As String

'define a variable to hold the number of records in the database
Public gintCountRecord As Integer

'a global variable to hold the game schedule, 2-dimensional array
'even colums contains Green house oponents
'odd columns contain Yellow house oponents
'Snap Green:0,Yellow:1
'Cribbage Green:2,Yellow:3
'Spllikins Green:4,Yellow:5
'Scrabble Green:6,Yellow:7
Public strGame(18, 8) As String

'a global variable to hold the green house studets names and classes,
'2-dimensional array
Public strGstudent(18, 2) As String

'a global variable to hold the yellow house students names and classes,
'2-dimensional array
Public strYstudent(18, 2) As String

'********************************************************************************
'a function that will open a connection with a database
'input: file name
'********************************************************************************
Public Sub DB_Connect(strFile As String)
    gstrConnection = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & _
```

```
mdlTOS - 2

        App.Path & "\" & strFile

    gadoConnection.Open gstrConnection
    Set gadoCommand.ActiveConnection = gadoConnection

End Sub

'*****************************************************************************
'a subroutine that will close the connection with a database
'*****************************************************************************
Public Sub DB_Disconnect()

    Set gadoRecordSet = Nothing
    Set gadoCommand = Nothing

    gadoConnection.Close
    Set gadoConnection = Nothing
End Sub

'*****************************************************************************
'a subroutine that reads the names of a specific house students
'input: house name:hs
'output: array of students names: str
'*****************************************************************************
Public Sub getStudentArray(str() As String, hs As String)

    Dim intCount As Integer
    Dim intX As Integer
    intX = 0
    'count number of students in hs
    gadoCommand.CommandText = _
            "SELECT COUNT (*) as recordc FROM Students WHERE House='" & hs & "'"
    Set gadoRecordSet = gadoCommand.Execute
    'store the count in intCount
    intCount = gadoRecordSet("recordc")
    'read first record
    gadoCommand.CommandText = _
        "SELECT * FROM Students WHERE House='" & hs & "'"
    Set gadoRecordSet = gadoCommand.Execute
    'store it
    str(intX, 0) = gadoRecordSet.Fields(1) 'name
    str(intX, 1) = gadoRecordSet.Fields(2) 'class

    'read the rest of the records
    For intX = 1 To intCount - 1
        gadoCommand.CommandText = _
            "SELECT * FROM Students WHERE House='" & hs & "'"
        Set gadoRecordSet = gadoCommand.Execute
        gadoRecordSet.GetRows (intX) 'get the record number intX
        str(intX, 0) = gadoRecordSet.Fields(1) 'name
        str(intX, 1) = gadoRecordSet.Fields(2) 'class
    Next intX

End Sub

'*****************************************************************************
'a subroutine that clears game information from the Access database
'*****************************************************************************
Public Sub clearDatabase()

    gadoCommand.CommandText = "UPDATE Students  Set PlayedSnap=false"
    Set gadoRecordSet = gadoCommand.Execute
    gadoCommand.CommandText = "UPDATE Students  Set SnapOponent=0"
    Set gadoRecordSet = gadoCommand.Execute
    gadoCommand.CommandText = "UPDATE Students  Set PlayedCribbage=false"
    Set gadoRecordSet = gadoCommand.Execute
    gadoCommand.CommandText = "UPDATE Students  Set CribbageOponent=0"
    Set gadoRecordSet = gadoCommand.Execute
    gadoCommand.CommandText = "UPDATE Students  Set PlayedScrabble=false"
    Set gadoRecordSet = gadoCommand.Execute
    gadoCommand.CommandText = "UPDATE Students  Set ScrabbleOponent=0"
    Set gadoRecordSet = gadoCommand.Execute
    gadoCommand.CommandText = "UPDATE Students  Set PlayedSpillikins=false"
    Set gadoRecordSet = gadoCommand.Execute
    gadoCommand.CommandText = "UPDATE Students  Set SpillikinsOponent=0"
```

```
mdlTOS - 3

    Set gadoRecordSet = gadoCommand.Execute

End Sub

'*****************************************************************************
'a function that checks if a specific student has played a specific game
'input: student name: str
'input: indicates gameand student house: intGameCol
'return true if str has played the game which col is intGameCol otherwise false
'*****************************************************************************
Public Function IsInGame(str As String, intGameCol As Integer) As Boolean

    Dim intX As Integer

    'compare the student name str to all students who played already if a match
    'is found true is returned and the search is terminated
    For intX = 0 To 17
        If strGame(intX, intGameCol) = str Then
            IsInGame = True
            Exit Function
        End If
    Next intX
    'if str did not match any of the students who already played
    'str did not play and a false is returned
    IsInGame = False

End Function

'*****************************************************************************
'a function that checks if two students form the yellow and green houses
'has played each other in any of the four games
'input: first student name: str1
'input: second student name: str2
'return true if str1 has already played str2
'*****************************************************************************
Public Function Played(str1 As String, str2 As String, intGameCol As Integer) As Boolean

    Dim intX As Integer
    Dim intY As Integer
    If intGameCol Mod 2 = 0 Then
        For intX = 0 To 7 'go through all games
            If intX Mod 2 = 0 Then
                For intY = 0 To 17 'check all matches
                'if the first student is found in the game schedule
                'and if the oponent is the second student-> str1 and str2
                'played each other and true is returned and the search is terminated
                    If strGame(intY, intX) = str1 Then
                        If strGame(intY, intX + 1) = str2 Then
                            Played = True
                            Exit Function
                        End If
                    End If
                Next intY
            End If
        Next intX
    Else
        For intX = 0 To 7 'go through all games
            If intX Mod 2 = 1 Then
                For intY = 0 To 17 'check all matches
                'if the first student is found in the game schedule
                'and if the oponent is the second student-> str1 and str2
                'played each other and true is returned and the search is terminated
                    If strGame(intY, intX) = str1 Then
                        If strGame(intY, intX - 1) = str2 Then
                            Played = True
                            Exit Function
                        End If
                    End If
                Next intY
            End If
        Next intX
    End If

    'if the search ended without finding matches false is returned
    Played = False
```

```
mdlTOS - 4


End Function

'*******************************************************************************
'a function that generate a list of all potential yellow house oponents for a
'student from the green house in a specific game
'input: student name: str
'input: indicates game and student house: intGameCol
'output: list of the names of the potential oponents
'return the number of potential oponents
'*******************************************************************************
Public Function potentialOponents(str As String, _
intGameCol As Integer, oponents() As String) As Integer

'integer to hold "str" class
    Dim intClass As Integer
    Dim intX As Integer
    'number of openents
    potentialOponents = 0
    'Clear oponents array
    For intX = LBound(oponents) To UBound(oponents)
        oponents(intX) = ""
    Next intX
    'game number is odd the yellow house student is in consideration
    'yellow
    If intGameCol Mod 2 = 1 Then
        'find the class the student is in
        For intX = 0 To 17
            If strGstudent(intX, 0) <> "" Then
                If strGstudent(intX, 0) = str Then
                    intClass = strGstudent(intX, 1)
                    Exit For
                End If
            End If
        Next intX
        'search through the list to find student who can play
        For intX = 0 To 17
            If strYstudent(intX, 0) <> "" Then
                If strYstudent(intX, 1) <> intClass Then
                    If Not IsInGame(strYstudent(intX, 0), intGameCol) Then
                        If Not Played(strYstudent(intX, 0), str, intGameCol) Then
                            oponents(potentialOponents) = strYstudent(intX, 0)
                            potentialOponents = potentialOponents + 1
                        End If
                    End If
                End If
            End If
       Next intX
    'green
    Else
         'find the class the student is in
      For intX = 0 To 17
           If strYstudent(intX, 0) <> "" Then
                If strYstudent(intX, 0) = str Then
                    intClass = strYstudent(intX, 1)
                    Exit For
                End If
            End If
       Next intX
        'search through the list
        For intX = 0 To 17
            If strGstudent(intX, 0) <> "" Then
                If strGstudent(intX, 1) <> intClass Then
                    If Not IsInGame(strGstudent(intX, 0), intGameCol) Then
                        If Not Played(strGstudent(intX, 0), str, intGameCol) Then
                            oponents(potentialOponents) = strGstudent(intX, 0)
                            potentialOponents = potentialOponents + 1
                        End If
                    End If
               End If
            End If
         Next intX
    End If

End Function
```

```
mdlTOS - 5

'*****************************************************************************
'a function that clears the game array
'*****************************************************************************
Public Sub clearGameArray()
    Dim intX As Integer
    Dim intY As Integer

    For intX = 0 To 7
        For intY = 0 To 17
            strGame(intY, intX) = ""
        Next intY
    Next intX

End Sub

'*****************************************************************************
'a function that generate a list of possible players added to a combo box
'input: the name of the game: strGame
'input: the combo box that will hold the potential players names
'output: a list of all potential players at that point will be entered into
'the combo box
'*****************************************************************************
Public Sub Manual(cmb1 As ComboBox, txtOp As String, intIndex As Integer _
, intGameCol As Integer)

    Dim op(18) As String
    Dim intOp As Integer
    Dim intX As Integer
    Dim strTemp As String
    'remove the current player from game
    If gboolEdit = True Then
        strTemp = ""
    Else
        strTemp = cmb1.Text
    End If

    'update the database (remove the player)
    gadoCommand.CommandText = "UPDATE Students Set PlayedCribbage=False " _
    & "where studentName='" & strTemp & "'"
    Set gadoRecordSet = gadoCommand.Execute
    strGame(intIndex, intGameCol) = ""

    cmb1.Clear
    'generate a list of all possible players in that place
    intOp = potentialOponents(txtOp, intGameCol, op())

    'add the players to the combo box list
    For intX = 0 To intOp - 1
        cmb1.AddItem op(intX)
    Next intX
End Sub

'*****************************************************************************
'a function that increments the number of matches played by a student in a game
'input: the name of the game: strGame
'input:the name of the student: strName
'the process is carried out on the database
'*****************************************************************************
Public Sub UpdateMatchCount(strName As String, strGame As String)

    'define a variable to hold the previous number of matches
    Dim intTempCount As Integer
     'get the previous value from the database
    gadoCommand.CommandText = "SELECT * FROM statistics" & _
        " WHERE StudentName='" & strName & "'"
    Set gadoRecordSet = gadoCommand.Execute

    'store the value depending on the game considered
    If strGame = "Snap" Then
        intTempCount = gadoRecordSet.Fields(1)
    ElseIf strGame = "Cribbage" Then
        intTempCount = gadoRecordSet.Fields(2)
    ElseIf strGame = "Spillikins" Then
        intTempCount = gadoRecordSet.Fields(3)
    ElseIf strGame = "Scrabble" Then
```

```
mdlTOS - 6

        intTempCount = gadoRecordSet.Fields(4)
    Else
        myMsgBox "Error", "Ok", "Error"
    End If
    'increment the games played and update the database
    intTempCount = intTempCount + 1
     gadoCommand.CommandText = "UPDATE statistics  Set " & strGame & "Matches=" _
       & intTempCount & " where studentName='" & strName & "'"
      Set gadoRecordSet = gadoCommand.Execute

End Sub

'*************************************************************************
'a function that increments the number of matches won by a student in a game
'it also store the max score the student achieved
'input: the name of the game: strGame
'input:the name of the student: strName
'input:the score of the student, intScore
'the process is carried out on the database
'*************************************************************************
Public Sub UpdateMatchWonCount(strName As String, strGame As String, _
intScore As Integer)
    'define a variable to hold the previous number of matches
    Dim intTempCount As Integer
    Dim intTempScore As Integer
    'get the previous value from the database
    gadoCommand.CommandText = "SELECT * FROM statistics" & _
        " WHERE StudentName='" & strName & "'"
    Set gadoRecordSet = gadoCommand.Execute

    'store the value depending on the game considered
    If strGame = "Snap" Then
        intTempCount = Val(gadoRecordSet.Fields(5))
        intTempScore = Val(gadoRecordSet.Fields(9))
    ElseIf strGame = "Cribbage" Then
        intTempCount = Val(gadoRecordSet.Fields(6))
        intTempScore = Val(gadoRecordSet.Fields(10))
    ElseIf strGame = "Spillikins" Then
        intTempCount = Val(gadoRecordSet.Fields(7))
        intTempScore = Val(gadoRecordSet.Fields(11))
    ElseIf strGame = "Scrabble" Then
        intTempCount = Val(gadoRecordSet.Fields(8))
        intTempScore = Val(gadoRecordSet.Fields(12))
    Else
        myMsgBox "Error", "Ok", "Error"
    End If

    'increment the games won and update the database
    intTempCount = Val(intTempCount) + 1
    gadoCommand.CommandText = "UPDATE statistics  Set Won" & strGame & "=" _
    & intTempCount & " where studentName='" & strName & "'"
    Set gadoRecordSet = gadoCommand.Execute

    'if current score is greater than the score stored in the database,store it
    If intScore > intTempScore Then
        gadoCommand.CommandText = "UPDATE statistics  Set " & strGame & "Max =" _
    & intScore & " where studentName='" & strName & "'"
    Set gadoRecordSet = gadoCommand.Execute
    End If

End Sub

'*************************************************************************
'a function that increments the number of compititions won by a specific house
'input: the name of the house: strWinner
'the process is carried out on the database
'*************************************************************************
Public Sub UpdateTournaments(strWinner As String)
  'define a variable to hold the previous number of tournaments won
    Dim intTempNumber As Integer

    'read the previous number
    gadoCommand.CommandText = "SELECT * FROM Tournaments" & _
        " WHERE House='" & strWinner & "'"
    Set gadoRecordSet = gadoCommand.Execute
```

mdlTOS - 7

```
    intTempNumber = gadoRecordSet.Fields(1)
    'increament the number
    intTempNumber = intTempNumber + 1

    'update the database
    gadoCommand.CommandText = "UPDATE Tournaments  Set CompititionsWon=" _
    & intTempNumber & " where House='" & strWinner & "'"
    Set gadoRecordSet = gadoCommand.Execute
End Sub

'*************************************************************************
'a public function use frmMsgBox to diplay a Message
'input: the message contetnts: strMessage
'input:indication of the buttons that should be on the message form: strOption
'input:the title or caption of the message form:strTitle
'*************************************************************************
Public Function myMsgBox(strMessage As String, strOption As String, _
strTitle As String) As Variant
    'pass the options to the global variable used in the message form
    msgResponse = strOption
    frmMsgBox.lblMessage.Caption = strMessage
    frmMsgBox.Caption = strTitle
    frmMsgBox.Show (1)
    'store the response of the user
    If msgResponse = "Yes" Then
        myMsgBox = vbYes
    ElseIf msgResponse = "No" Then
        myMsgBox = vbNo
    ElseIf msgResponse = "OK" Then
        myMsgBox = vbOK
    ElseIf msgResponse = "Cancel" Then
        myMsgBox = vbCancel
    End If
End Function
```

5.2 Splash Form

5.2.1 User Interface (Task4/a)

This form is the first form to be loaded. It was not in the original design but I added it to give the programme a professional look. It displays the name of the system, the organisation using it, the programmes version, the name of the programmer or the developing company and the copy right information. Splash forms come as a template in Visual Basic 6. I used that but I customised the look to synchronise with the rest of the programme.

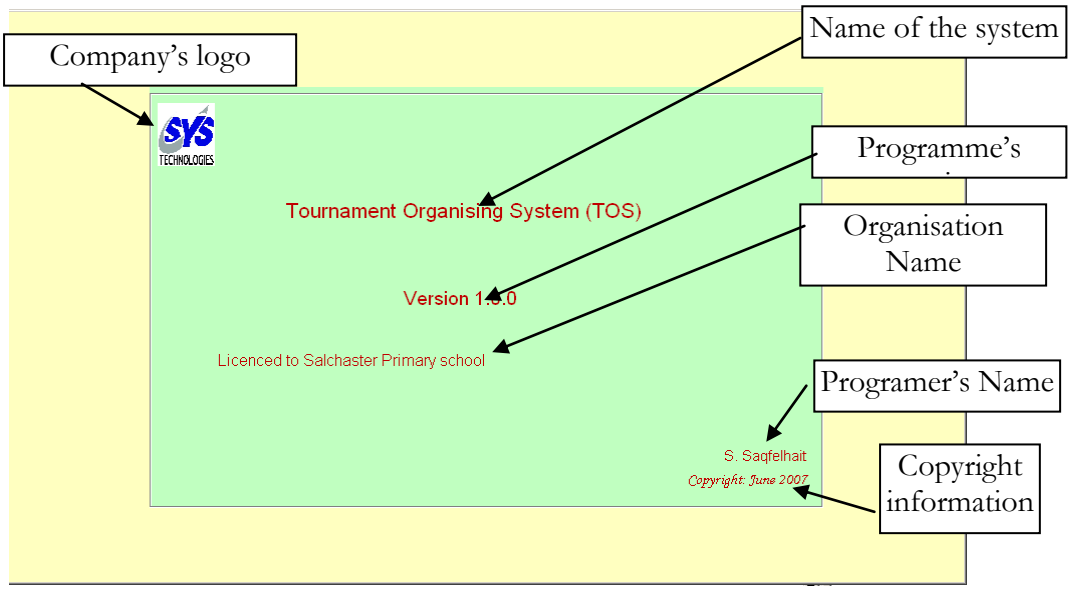


Figure 34

5.2.2 Testing (Task4/b)

Test no.	Test	Test data	Expected Result	Actual Results	Comments
1	Click form	-	Load login form	login form was loaded	Test passed
2	Click frame	-	Load login form	login form was loaded	Test passed
3	Do nothing	-	After 5 sec load login form	After 5 sec login form was loaded	Test passed

Table 16

5.2.3 Code (frmSplash.frm) (Task4/a)

frmSplash - 1

```
'**************************************************************************************
'************************** Tournament Organising System**********************************
'*********************************frmSplash Code*****************************************
'*******************************Programer: S. Saqfelhait**********************************
'*********************************Date:07/04/2007*****************************************
'**************************************************************************************
'startup form
Option Explicit

'**************************************************************************************
'subroutine,executed if any key is pressed
'input: the operating system pass the ascii value of the key pressed
'**************************************************************************************
Private Sub Form_KeyPress(KeyAscii As Integer)
    'if a key is pressed the introfrm will be shown
    '& the form will be unloaded
    frmLogin.Show
    Unload Me
End Sub

'**************************************************************************************
'subroutine will be executed when the form is loaded
'**************************************************************************************
Private Sub Form_Load()
    'set the timer to 5 seconds
    tmrSplash.Interval = 5000
    'enable the timer control
    tmrSplash.Enabled = True
    'allow the operating system to deal with other events other than the timeout
    DoEvents
    lblVersion.Caption = "Version " & App.Major & "." & _
        App.Minor & "." & App.Revision
    lblProductName.Caption = "Tournament Organising System (TOS)"
    lblLicenseTo.Caption = " Licenced to Salchaster Primary school"
    lblCompany.Caption = " S. Saqfelhait"
End Sub

'**************************************************************************************
'subroutine will be executed when frame1 is clicked
'**************************************************************************************
Private Sub Frame1_Click()
    frmLogin.Show
    Unload Me
End Sub

'**************************************************************************************
'subroutine will be executed when the timer is enabled and every  "interval"
'**************************************************************************************
Private Sub tmrAnimate_Timer()
'toggle form and frame colours
    If Me.BackColor = &HC0FFC0 Then
        Me.BackColor = &HC0FFFF
    Else
        Me.BackColor = &HC0FFC0
    End If

    If Frame1.BackColor = &HC0FFC0 Then
        Frame1.BackColor = &HC0FFFF
    Else
        Frame1.BackColor = &HC0FFC0
    End If
End Sub

'**************************************************************************************
'subroutine will be executed when the timer is enabled and every  "interval"
'**************************************************************************************
Private Sub tmrSplash_Timer()
    'this subroutine is exceuted after _
    'the timer is enabled by the interval value
    frmLogin.Show
    Unload Me
    tmrSplash.Enabled = False
End Sub
```

5.3 Login form

5.3.1 User interface (Task4/a)

This form is loaded when the splash screen is clicked, a key is pressed or the time specified for the splash screen display elapsed. Pressing cancel will end the programme. Entering the correct combination of username and password will load the main window form. It will also establish a connection to the access database.

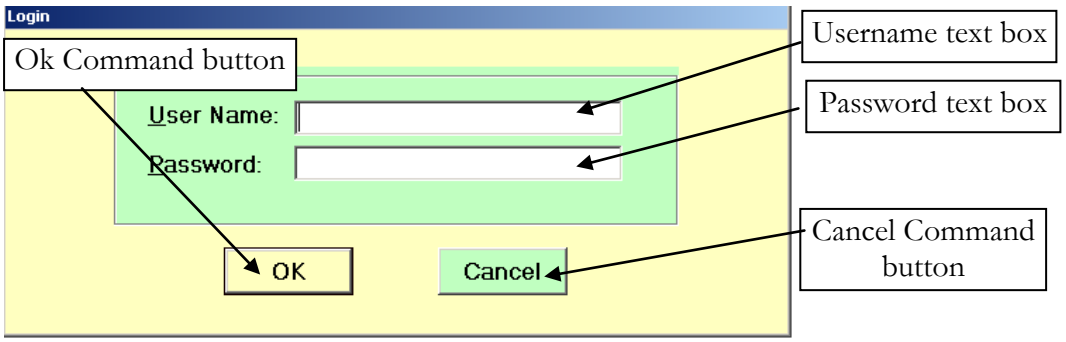


Figure 35

5.3.2 Testing (Task4/b)

To test each control, the correct data will be entered in all controls except the one that we are testing.

The username is: admin (not case sensitive) and the password is: aloha (case sensitive)

Test no.	Test	Test data	Expected Result	Actual Results	Comments
1	Username text box	blank	Error message asking for correct username	Error message popped up asking for correct username	Test passed
2	Username text box	admini	Error message asking for correct username	Error message popped up asking for correct username	Test passed
3	Username text box	Admin	Load the main window form	Error message popped up asking for correct username	Check the if statement and convert the input to upper case before comparing it
4	Username text box	admin	Load the main window form	Load the main window form	Test passed
5	Password text box	blank	Error message asking for correct password	Error message popped up asking for correct password	Test passed
6	Password text box	hawai	Error message asking for correct password	Error message popped up asking for correct password	Test passed
7	Password text box	Aloha	Error message popped up asking for correct password	Error message popped up asking for correct password	Test passed
8	Password text box	aloha	Load the main window form	Load the main window form	Test passed
9	Ok command button	Correct name and password	Load the main window form	Load the main window form	Test passed
10	Cancel command button	-	Exit the programme	The programme was terminated	Test passed

Table 17

5.3.3 Code (frmLogin.frm)(Task4/a)

```
frmLogin - 1

'************************** Tournament Organising System**********************
'*********************************frmLogin Code*******************************
'****************************Programer: Somoud Saqfelhait**********************
'*********************************Date:07/04/2007*****************************
'*****************************************************************************
'this is the login form loaded from the frmSplash form
Option Explicit

'*****************************************************************************
'subroutine,executed when the Cancel button is clicked
'*****************************************************************************
Private Sub cmdCancel_Click()
    End
End Sub

'*****************************************************************************
'subroutine,executed when the OK button is clicked
'*****************************************************************************
Private Sub cmdOk_Click()
    'check for username and correct password
    If UCase(txtUserName.Text) = "ADMIN" Then
        If txtPassword = "aloha" Then
            DB_Connect ("students.mdb")
            frmMain.Show
            Unload Me
        Else
            myMsgBox "Invalid Password, try again!", "OK", "Login"
            txtPassword.SetFocus
            SendKeys "{Home}+{End}"
        End If
    Else
        myMsgBox "Invalid UserName, try again!", "OK", "Login"
        txtUserName.SetFocus
            SendKeys "{Home}+{End}"
    End If
End Sub
```

5.4 Main Form

5.4.1 User interface (Task4/a)

This window is the main window of the programme. It is loaded when the correct combination of the username and password are entered in the login form and the Ok command button is clicked.

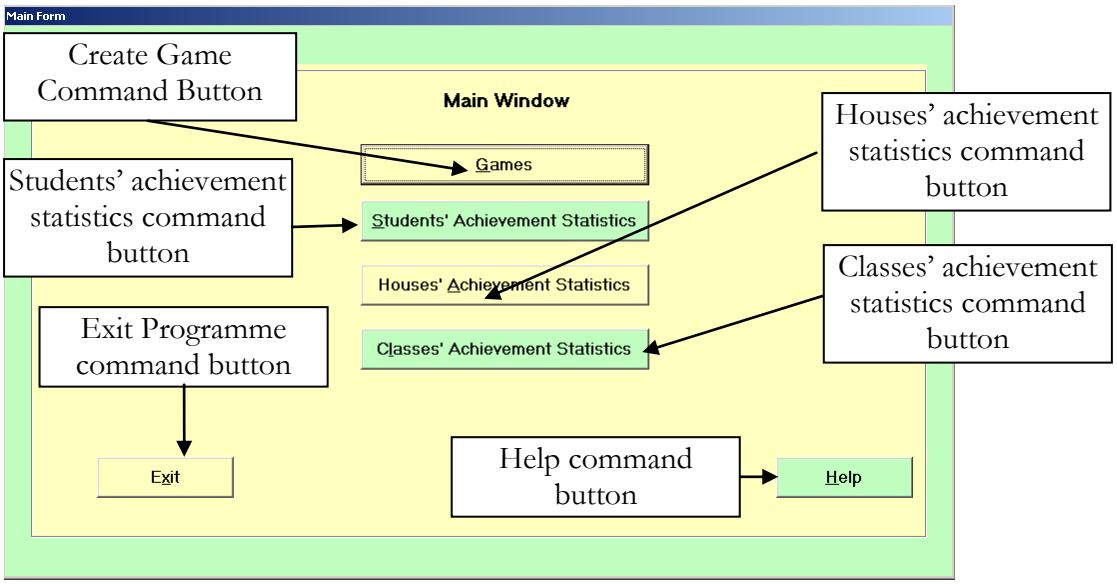


Figure 36

5.4.2 Testing (Task4/b)

To test each control, the correct data will be entered in all controls except the one that we are testing.

Test no.	Test	Test data	Expected Result	Actual Results	Comments
1	Games Button	-	The create game form will be loaded	The create game form was loaded	Test passed
2	Students' achievement statistics button	-	The students statistics form will be loaded	The students statistics form was loaded	Test passed
3	Houses' achievement statistics button	-	The houses statistics form will be loaded	The students statistics form was loaded	Test passed
4	Classes' achievement statistics button	-	The classes statistics form will be loaded	The classes statistics form was loaded	Test passed
5	Exit command button	-	A message box will be displayed to ask the user to confirm if he/she want to exit the programme	A message box is displayed asking to confirm	Test passed
6	Help command button	-	The students statistics form will be loaded	The students statistics form will be loaded	Test passed

Table 18

5.4.3 Code (frmMain.frm)(Task4/a)

```
frmMain - 1

'************************** Tournament Organising System***********************
'*********************************frmMain Code*********************************
'****************************Programer: Somoud Saqfelhait************************
'*********************************Date:07/04/2007*****************************
'*******************************************************************************
'this is the main menu form loaded on successful login
Option Explicit

'*******************************************************************************
'subroutine,executed when the Classes Achievment Statistics button is clicked
'*******************************************************************************
Private Sub cmdstClasses_Click()
    frmClassSt.Show
    Unload Me
End Sub
'*******************************************************************************
'subroutine,executed when the Houses Achievment Statistics button is clicked
'*******************************************************************************
Private Sub cmdstHouse_Click()
    frmHouseSt.Show
    Unload Me
End Sub
'*******************************************************************************
'this subroutine is exceuted after the timer is enabled by the interval value
'*******************************************************************************
Private Sub tmrAnimate_Timer()
    'toggle form and frame colours
    If Me.BackColor = &HC0FFC0 Then
        Me.BackColor = &HC0FFFF
    Else
        Me.BackColor = &HC0FFC0
    End If
    If Frame1.BackColor = &HC0FFC0 Then
        Frame1.BackColor = &HC0FFFF
    Else
        Frame1.BackColor = &HC0FFC0
    End If
End Sub

'*******************************************************************************
'subroutine,executed when the Games button is clicked
'*******************************************************************************
Private Sub cmdCreate_Click()
    frmIntro.Show
    Unload Me
End Sub
'*******************************************************************************
'subroutine,executed when the Exit button is clicked
'*******************************************************************************
Private Sub cmdExit_Click()
   'define a variable to capture the msgbox response
    Dim varResponce As Variant
    varResponce = myMsgBox("Are You Sure you Want to Exit", "YesNo", "Attention")
    'exit program if response if Yes
    If varResponce = vbYes Then
        DB_Disconnect
        End
    End If
End Sub

'*******************************************************************************
'subroutine,executed when the Help button is clicked
'*******************************************************************************
Private Sub cmdHelp_Click()
    frmHelp.Show (1)
End Sub

'*******************************************************************************
'subroutine,executed when the Students Achievment Statistics button is clicked
'*******************************************************************************
Private Sub cmdStatistics_Click()
    frmStatistics.Show
    Unload Me
End Sub
```

5.5 Help Form

5.5.1 User interface (Task4/a)

This form is loaded from the Main Window when the help command button is clicked. It will display information to help the user use the programme. The form is composed of a label and a command button. The information displayed is read from a text file store in the same directory as the programme. (help.txt). this form was not in the original design but it is important so the users could know how to use the programme

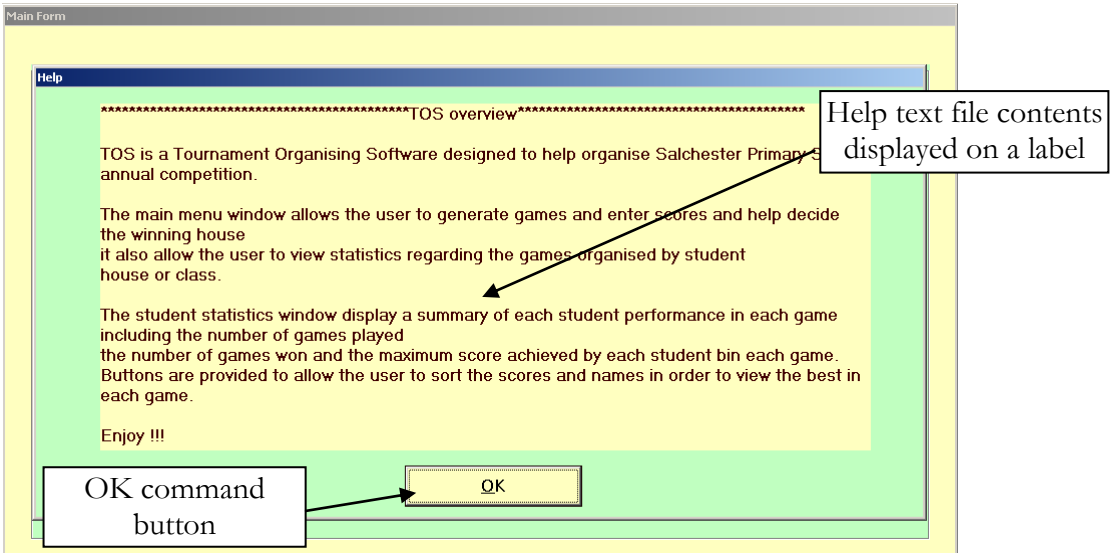


Figure 37

5.5.2 Testing (Task4/b)

As shown in figure 37 the help file is shown

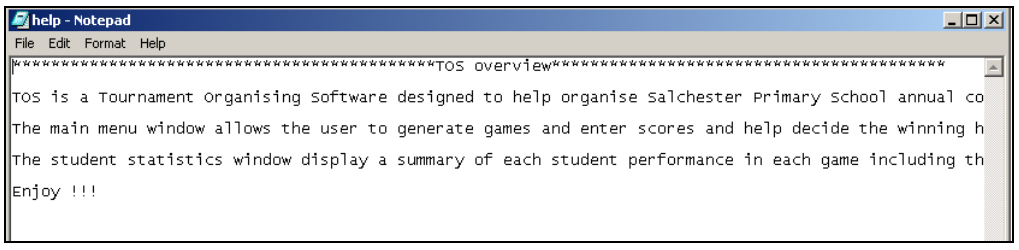


Figure 38

5.5.3 Code (frmHelp.frm) (Task4/a)

```
frmHelp - 1

'*************************Tournament Organising System*************************
'*********************************frmHelp Code*********************************
'*******************************Programer: S. Saqfelhait***********************
'*********************************Date:07/06/2006******************************
'******************************************************************************
'this form is loaded from the main menu form when the help button is clicked
Option Explicit

'******************************************************************************
'this subroutine is executed when the OK button is clicked
'******************************************************************************
Private Sub cmdOk_Click()
    Unload Me
End Sub
'******************************************************************************
'this subroutine is executed when the form is loaded
'******************************************************************************
Private Sub Form_Load()
    Dim strFileName As String
    Dim intX As Integer
    Dim strHelpText As String

    'get the full path of the help file
    strFileName = App.Path & "\help.txt"
    'open the help file in read mode
    Open strFileName For Input As #1
    'read first line
    Input #1, strHelpText
    lblHelp.Caption = strHelpText
    'read all the lines of the file until the end of the file is reached
    Do While Not EOF(1)
        Input #1, strHelpText
        lblHelp.Caption = lblHelp.Caption & vbCrLf & strHelpText
    Loop
    Close #1
End Sub
```

5.6 Students' Achievement Statistics Form

5.6.1 User interface (Task4/a)

This form is loaded from the main window when the Students' Achievements Statistics command button is clicked. It displays statistics about students in the game. It displays name of student, the number of Snap, Cribbage, Spillikins and Scrabble games he/she played, the number of Snap, Cribbage, Spillikins and Scrabble games he/she won, the maximum score he/she achieved in each game. Clicking Ok will return to the Main Window. Each student record back colour is set to the colour of the house they belong to.

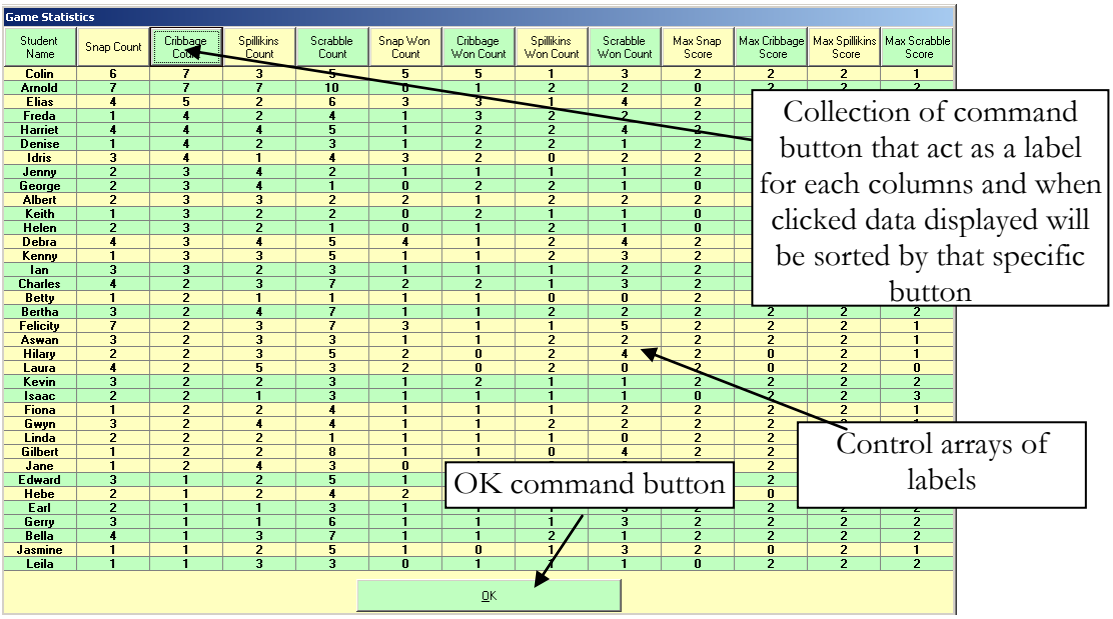


Figure 39

The results of each student are updated each time the games' results are entered and the winning house is decided upon.

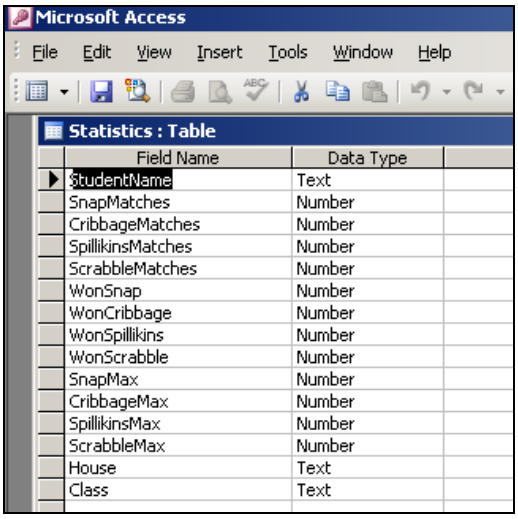


Figure 40(Access table used to store statistics/design view)

The function code used to sort data according to button pressed is shown below

```
'*************************************************************************
'subroutine, read student statistics  from the database
'and sort specific field depending on the value of mstrOrder
'input: the attribute to be sorted:strAttr
'*************************************************************************
Private Sub sortStatistics(strAttr As String)
 Dim intX As Integer
 'hold the colour of a specific introw of label boxes label box
 Dim colour As Variant
'read the  records
    For intX = 1 To 35
    'mstrOrder is a module level variable with values "ASC" or "DESC"
        gadoCommand.CommandText = "SELECT * FROM statistics ORDER BY " _
        & strAttr & " " & mstrOrder
        Set gadoRecordSet = gadoCommand.Execute
        gadoRecordSet.GetRows (intX)
        lblName(intX).Caption = gadoRecordSet.Fields(0)
        lblSnapCount(intX).Caption = gadoRecordSet.Fields(1)
        lblCribbageCount(intX).Caption = gadoRecordSet.Fields(2)
        lblSpillikinsCount(intX).Caption = gadoRecordSet.Fields(3)
        lblScrabbleCount(intX).Caption = gadoRecordSet.Fields(4)
        lblWonSnap(intX).Caption = gadoRecordSet.Fields(5)
        lblWonCribbage(intX).Caption = gadoRecordSet.Fields(6)
        lblWonSpillikins(intX).Caption = gadoRecordSet.Fields(7)
        lblWonScrabble(intX).Caption = gadoRecordSet.Fields(8)
        lblSnapMax(intX).Caption = gadoRecordSet.Fields(9)
        lblCribbageMax(intX).Caption = gadoRecordSet.Fields(10)
        lblSpillikinsMax(intX).Caption = gadoRecordSet.Fields(11)
        lblScrabbleMax(intX).Caption = gadoRecordSet.Fields(12)

        If gadoRecordSet.Fields(13) = "Yellow" Then
            colour = &HC0FFFF
        Else
            colour = &HC0FFC0
        End If
        Call setColour(intX, colour)
    Next intX
End Sub
```

Note: the three statistics forms were only one form in the original design but I found it easier to change to three forms and use labels ”control arrays” instead of list box(see the design in figure28/page24).

5.6.2 Testing (Task4/b)

To test each control, the correct data will be entered in all controls except the one that we are testing.

Test no.	Test	Test data	Expected Result	Actual Results	Comments
1	Ok Command Button	-	The main window will be loaded	The main window was loaded	Test passed
2	Student Name button	-	The records should be sorted by alphabetical student names order	The records was sorted by alphabetical student names order	Test passed
3	Snap Count button	-	The records should be sorted by descending snap count numbers	The records was sorted by descending snap count numbers	Test passed
4	Cribbage Count button	-	The records should be sorted by descending cribbage count numbers	The records was sorted by descending cribbage count numbers	Test passed
5	Spillikins Count button	-	The records should be sorted by descending Spillikins count numbers	The records was sorted by descending Spillikins count numbers	Test passed
6	Scrabble Count button	-	The records should be sorted by descending scrabble count numbers	The records was sorted by descending scrabble count numbers	Test passed
7	Snap Won count button	-	The records should be sorted by descending snap won counts	The records was sorted by descending snap won counts	Test passed
8	Cribbage Won count button	-	The records should be sorted by descending cribbage won counts	The records was sorted by descending cribbage won counts	Test passed
9	Spillikins Won count button	-	The records should be sorted by descending Spillikins won counts	The records was sorted by descending Spillikins won counts	Test passed
10	Scrabble Won count button	-	The records should be sorted by descending scrabble won counts	The records was sorted by descending scrabble won counts	Test passed
11	Snap Max Score command button	-	The records should be sorted by descending snap maximum score	The records was sorted by descending snap maximum score	Test passed
12	Cribbage Max Score command button	-	The records should be sorted by descending cribbage maximum score	The records was sorted by descending cribbage maximum score	Test passed
13	Spillikins Max Score command button	-	The records should be sorted by descending Spillikins maximum score	The records was sorted by descending Spillikins maximum score	Test passed
14	Scrabble Max Score command button	-	The records should be sorted by descending scrabble maximum score	The records was sorted by descending scrabble maximum score	Test passed

Table 19

5.6.3 Code (frmStatistics.frm)(task4/a)

```
frmStatistics - 1

'************************** Tournament Organising System**********************
'*********************************frmHouseSt Code*****************************
'****************************Programer: Somoud Saqfelhait**********************
'*********************************Date:07/06/2007*****************************
'*****************************************************************************
'this form will be loaded from the main menu form when House Achievments
'Statistics  command button is clicked

Option Explicit
'a module level variable which holds the sorting (Ascending or Descending)
Dim mstrOrder As String
'a module level variable which holds the last command button clicked
Dim mstrButton As String

'*****************************************************************************
'subroutine will be executed when the Cribbage Count command button is clicked
'*****************************************************************************
Private Sub cmdCribbageCount_Click()
    'if this button was the last to be clicked reverse the sorting
    If mstrButton = "cmdCribbageCount" Then
        If mstrOrder = "ASC" Then
            mstrOrder = "DESC"
        Else
            mstrOrder = "ASC"
        End If
    'order by maximum
    Else
        mstrOrder = "DESC"
    End If
    Call sortStatistics("CribbageMatches")
    mstrButton = "cmdCribbageCount"
End Sub

'*****************************************************************************
'subroutine will be executed when the Cribbage Max Score command button is clicked
'*****************************************************************************
Private Sub cmdCribbageMax_Click()
    'if this button was the last to be clicked reverse the sorting
    If mstrButton = "cmdCribbageMax" Then
        If mstrOrder = "ASC" Then
            mstrOrder = "DESC"
        Else
            mstrOrder = "ASC"
        End If
    'order by maximum
    Else
        mstrOrder = "DESC"
    End If
    Call sortStatistics("CribbageMax")
    mstrButton = "cmdCribbageMax"
End Sub

'*****************************************************************************
'subroutine will be executed when the Ok command button is clicked
'*****************************************************************************
Private Sub cmdOk_Click()
    frmMain.Show
    Unload Me
End Sub

'*****************************************************************************
'subroutine will be executed when the Scrabble Count command button is clicked
'*****************************************************************************
Private Sub cmdScrabbleCount_Click()
    'if this button was the last to be clicked reverse the sorting
    If mstrButton = "cmdScrabbleCount" Then
        If mstrOrder = "ASC" Then
            mstrOrder = "DESC"
        Else
            mstrOrder = "ASC"
        End If
```

frmStatistics - 2

```
    'order by maximum
    Else
        mstrOrder = "DESC"
    End If
    Call sortStatistics("ScrabbleMatches")
    mstrButton = "cmdScrabbleCount"
End Sub

'*************************************************************************
'subroutine will be executed when the Scrabble Max Score command button is clicked
'*************************************************************************
Private Sub cmdScrabbleMax_Click()
    'if this button was the last to be clicked reverse the sorting
    If mstrButton = "cmdScrabbleMax" Then
        If mstrOrder = "ASC" Then
            mstrOrder = "DESC"
        Else
            mstrOrder = "ASC"
        End If
    Else
        'order by maximum
        mstrOrder = "DESC"
    End If
    Call sortStatistics("ScrabbleMax")
    mstrButton = "cmdScrabbleMax"
End Sub

'*************************************************************************
'subroutine will be executed when the Snap Count command button is clicked
'*************************************************************************
Private Sub cmdSnapCount_Click()
    'if this button was the last to be clicked reverse the sorting
    If mstrButton = "cmdSnapCount" Then
        If mstrOrder = "ASC" Then
            mstrOrder = "DESC"
        Else
            mstrOrder = "ASC"
        End If
    Else
        'order by max
        mstrOrder = "DESC"
    End If
    Call sortStatistics("SnapMatches")
    mstrButton = "cmdSnapCount"
End Sub

'*************************************************************************
'subroutine will be executed when the Snap Max Score command button is clicked
'*************************************************************************
Private Sub cmdSnapMax_Click()
    'if this button was the last to be clicked reverse the sorting
    If mstrButton = "cmdSnapMax" Then
        If mstrOrder = "ASC" Then
            mstrOrder = "DESC"
        Else
            mstrOrder = "ASC"
        End If
    Else
        mstrOrder = "DESC"
    End If
    Call sortStatistics("SnapMax")
    mstrButton = "cmdSnapMax"

End Sub

'*************************************************************************
'subroutine will be executed when the Spillikins Count command button is clicked
'*************************************************************************
Private Sub cmdSpillikinsCount_Click()
    'if this button was the last to be clicked reverse the sorting
    If mstrButton = "cmdSpillikinsCount" Then
        If mstrOrder = "ASC" Then
            mstrOrder = "DESC"
        Else
            mstrOrder = "ASC"
```

```
frmStatistics - 3

        End If
    Else
        mstrOrder = "DESC"
    End If
     Call sortStatistics("SpillikinsMatches")
    mstrButton = "cmdSpillikinsCount"

End Sub

'*************************************************************************
'subroutine will be executed when the Spillikins Max Score command button is clicked
'*************************************************************************
Private Sub cmdSpillikinsMax_Click()
    'if this button was the last to be clicked reverse the sorting
    If mstrButton = "cmdSpillikinsMax" Then
        If mstrOrder = "ASC" Then
            mstrOrder = "DESC"
        Else
            mstrOrder = "ASC"
        End If
    Else
        mstrOrder = "DESC"
    End If
    Call sortStatistics("SpillikinsMax")
    mstrButton = "cmdSpillikinsMax"

End Sub

'*************************************************************************
'subroutine will be executed when the Student name command button is clicked
'*************************************************************************
Private Sub cmdStudentName_Click()
    'if this button was the last to be clicked reverse the sorting
    If mstrButton = "cmdStudentName" Then
        If mstrOrder = "ASC" Then
            mstrOrder = "DESC"
        'order alphabetically starting from a-z
        Else
            mstrOrder = "ASC"
        End If
    Else
        mstrOrder = "ASC"
    End If
    Call sortStatistics("StudentName")
    mstrButton = "cmdStudentName"
End Sub

'*************************************************************************
'subroutine will be executed when the Cribbage Won Count command button is clicked
'*************************************************************************
Private Sub cmdWonCribbage_Click()
    'if this button was the last to be clicked reverse the sorting
    If mstrButton = "cmdWonCribbage" Then
        If mstrOrder = "ASC" Then
            mstrOrder = "DESC"
        Else
            mstrOrder = "ASC"
        End If
    Else
        mstrOrder = "DESC"
    End If
    Call sortStatistics("WonCribbage")
    mstrButton = "cmdWonCribbage"

End Sub

'*************************************************************************
'subroutine will be executed when the Scrabble Won Count command button is clicked
'*************************************************************************
Private Sub cmdWonScrabble_Click()
    'if this button was the last to be clicked reverse the sorting
    If mstrButton = "cmdWonScrabble" Then
        If mstrOrder = "ASC" Then
            mstrOrder = "DESC"
        Else
```

frmStatistics - 4

```
            mstrOrder = "ASC"
        End If
    Else
        mstrOrder = "DESC"
    End If
    Call sortStatistics("WonScrabble")
    mstrButton = "cmdWonScrabble"

End Sub

'*************************************************************************
'subroutine will be executed when the Snap Won count command button is clicked
'*************************************************************************
Private Sub cmdWonSnap_Click()
    'if this button was the last to be clicked reverse the sorting
    If mstrButton = "cmdWonSnap" Then
        If mstrOrder = "ASC" Then
            mstrOrder = "DESC"
        Else
            mstrOrder = "ASC"
        End If
    Else
        mstrOrder = "DESC"
    End If
    Call sortStatistics("WonSnap")
    mstrButton = "cmdWonSnap"

End Sub

'*************************************************************************
'subroutine will be executed when the Spillikins Won Countcommand button is clicked
'*************************************************************************
Private Sub cmdWonSpillikins_Click()
    'if this button was the last to be clicked reverse the sorting
    If mstrButton = "cmdWonSpillikins" Then
        If mstrOrder = "ASC" Then
            mstrOrder = "DESC"
        Else
            mstrOrder = "ASC"
        End If
    Else
        mstrOrder = "DESC"
    End If
    Call sortStatistics("WonSpillikins")
    mstrButton = "cmdWonSpillikins"

End Sub

'*************************************************************************
'subroutine will be executed when the form is loaded
'*************************************************************************
Private Sub Form_Load()
    mstrOrder = "ASC"
    Call sortStatistics("StudentName")
    mstrButton = "cmdStudentName"
End Sub
'*************************************************************************
'subroutine, read student statistics  from the database
'and sort specific field depending on the value of mstrOrder
'input: the attribute to be sorted:strAttr
'*************************************************************************
Private Sub sortStatistics(strAttr As String)
 Dim intX As Integer
 'hold the colour of a specific introw of label boxes label box
 Dim colour As Variant

 'read the first record
    gadoCommand.CommandText = "SELECT * FROM statistics ORDER BY " _
    & strAttr & " " & mstrOrder
        Set gadoRecordSet = gadoCommand.Execute
        'dispaly reord
        lblName(0).Caption = gadoRecordSet.Fields(0)
        lblSnapCount(0).Caption = gadoRecordSet.Fields(1)
        lblCribbageCount(0).Caption = gadoRecordSet.Fields(2)
        lblSpillikinsCount(0).Caption = gadoRecordSet.Fields(3)
```

frmStatistics - 5

```
        lblScrabbleCount(0).Caption = gadoRecordSet.Fields(4)
        lblWonSnap(0).Caption = gadoRecordSet.Fields(5)
        lblWonCribbage(0).Caption = gadoRecordSet.Fields(6)
        lblWonSpillikins(0).Caption = gadoRecordSet.Fields(7)
        lblWonScrabble(0).Caption = gadoRecordSet.Fields(8)
        lblSnapMax(0).Caption = gadoRecordSet.Fields(9)
        lblCribbageMax(0).Caption = gadoRecordSet.Fields(10)
        lblSpillikinsMax(0).Caption = gadoRecordSet.Fields(11)
        lblScrabbleMax(0).Caption = gadoRecordSet.Fields(12)

        'change colours of the labels based on student's house colour
        If gadoRecordSet.Fields(13) = "Yellow" Then
            colour = &HC0FFFF
        Else
            colour = &HC0FFC0
        End If
        'change the colours
        Call setColour(0, colour)
'read the remaining records
    For intX = 1 To 35

        gadoCommand.CommandText = "SELECT * FROM statistics ORDER BY " _
        & strAttr & " " & mstrOrder
        Set gadoRecordSet = gadoCommand.Execute
        gadoRecordSet.GetRows (intX)
        lblName(intX).Caption = gadoRecordSet.Fields(0)
        lblSnapCount(intX).Caption = gadoRecordSet.Fields(1)
        lblCribbageCount(intX).Caption = gadoRecordSet.Fields(2)
        lblSpillikinsCount(intX).Caption = gadoRecordSet.Fields(3)
        lblScrabbleCount(intX).Caption = gadoRecordSet.Fields(4)
        lblWonSnap(intX).Caption = gadoRecordSet.Fields(5)
        lblWonCribbage(intX).Caption = gadoRecordSet.Fields(6)
        lblWonSpillikins(intX).Caption = gadoRecordSet.Fields(7)
        lblWonScrabble(intX).Caption = gadoRecordSet.Fields(8)
        lblSnapMax(intX).Caption = gadoRecordSet.Fields(9)
        lblCribbageMax(intX).Caption = gadoRecordSet.Fields(10)
        lblSpillikinsMax(intX).Caption = gadoRecordSet.Fields(11)
        lblScrabbleMax(intX).Caption = gadoRecordSet.Fields(12)

        If gadoRecordSet.Fields(13) = "Yellow" Then
            colour = &HC0FFFF
        Else
            colour = &HC0FFC0
        End If
        Call setColour(intX, colour)
    Next intX
End Sub

'*************************************************************************
'subroutine, change the colour of specific introw of labels to varClr
'input: the number of introw of labels  to be changed:intRow
'input:the desired colour :varClr
'*************************************************************************
Private Sub setColour(intRow As Integer, varClr As Variant)
        lblName(intRow).BackColor = varClr
        lblSnapCount(intRow).BackColor = varClr
        lblCribbageCount(intRow).BackColor = varClr
        lblSpillikinsCount(intRow).BackColor = varClr
        lblScrabbleCount(intRow).BackColor = varClr
        lblWonSnap(intRow).BackColor = varClr
        lblWonCribbage(intRow).BackColor = varClr
        lblWonSpillikins(intRow).BackColor = varClr
        lblWonScrabble(intRow).BackColor = varClr
        lblSnapMax(intRow).BackColor = varClr
        lblCribbageMax(intRow).BackColor = varClr
        lblSpillikinsMax(intRow).BackColor = varClr
        lblScrabbleMax(intRow).BackColor = varClr
End Sub
```

5.7 Houses' Achievement Statistics Form

5.7.1 User interface (Task4/a)

This form is loaded from the main window when the Houses' Achievements Statistics command button is clicked. It displays statistics about houses in the game.

The information displayed about each house are read and calculated from the database using SQL commands, as shown in the example below.

```
'*************************************************************************
'a function that calculate the sum of a specifi field in the database
'input: Class number: intClass
'input: the attribute location within the database (column position):intCol
'return the sum of the field
'*************************************************************************
Private Function Statistics(strHouse As String, intCol As Integer) As Integer
    Dim intX As Integer
    'integer value used to accumulate the values of the col
    Dim intCount As Integer

    'get the first record
    gadoCommand.CommandText = "SELECT * FROM Statistics WHERE House= '" _
    & strHouse & "'"
    Set gadoRecordSet = gadoCommand.Execute
     'intialise the accumulator
    intCount = Val(gadoRecordSet.Fields(intCol))

    'read the rest of the records
    For intX = 1 To 35
        gadoCommand.CommandText = "SELECT * FROM Statistics WHERE House= '" _
        & strHouse & "'"
        Set gadoRecordSet = gadoCommand.Execute
        gadoRecordSet.GetRows (intX)
        'if no more records belong to intClass exit the for loop
        If gadoRecordSet.EOF = True Then Exit For
        'add the current record field value to the accumulator
        intCount = intCount + Val(gadoRecordSet.Fields(intCol))
    Next intX
    'return the result
    Statistics = intCount
End Function
```

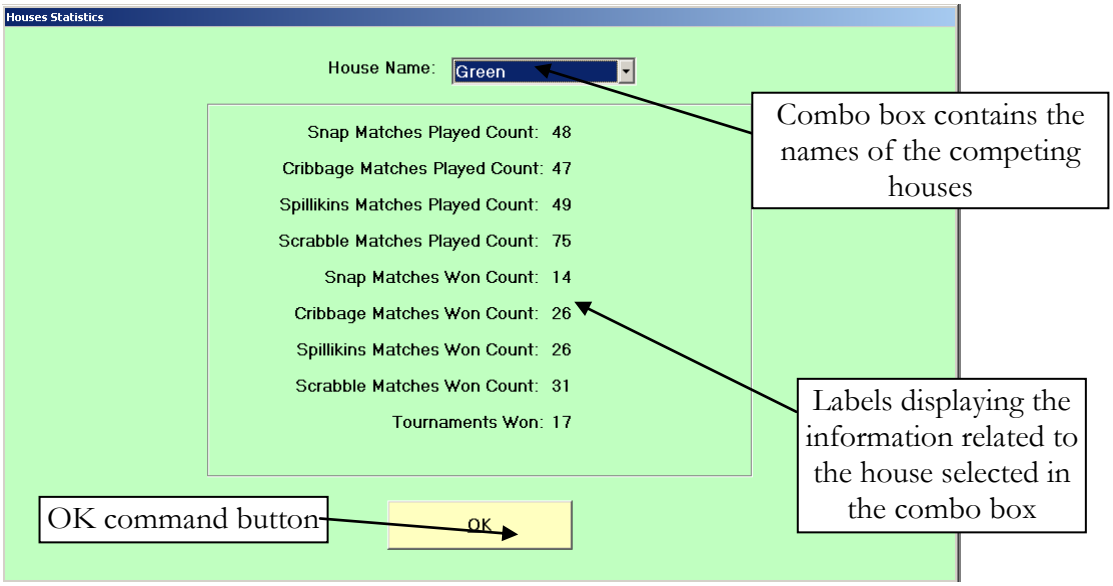


Figure 41

5.7.2 Testing (Task4/b)

To test each control, the correct data will be entered in all controls except the one that we are testing.

Test no.	Test	Test data	Expected Result	Actual Results	Comments
1	Ok Command Button	-	The main window will be loaded	The main window was loaded	Test passed
2	House name combo box	Green	The information about the green house will be displayed and the background will change to green colour	The information about the green house was displayed and the background changed to green colour	Test passed
3	House name combo box	Yellow	The information about the yellow house will be displayed and the background will change to yellow colour	The information about the yellow house was displayed and the background changed to yellow colour	Test passed

Table 20

5.7.3 Code (frmHouseSt.frm) (Task4/a)

```
frmHouseSt - 1

'************************* Tournament Organising System***********************
'*********************************frmHouseSt Code*****************************
'****************************Programer: Somoud Saqfelhait***********************
'*********************************Date:07/06/2007*****************************
'*****************************************************************************
'this form will be loaded from the main menu form when House Achievments
'Statistics  command button is clicked
Option Explicit

'*****************************************************************************
'subroutine will be executed when the house combo box is clicked to select item
'*****************************************************************************
Private Sub cmbHouse_Click()

    If cmbHouse.ListIndex <> -1 Then
        lblSnapCount.Caption = ""
        lblCribbageCount.Caption = ""
        lblSpillikinsCount.Caption = ""
        lblScrabbleCount.Caption = ""
        lblSnapWon.Caption = ""
        lblCribbageWon.Caption = ""
        lblSpillikinsWon.Caption = ""
        lblScrabbleWon.Caption = ""
        lblTournaments.Caption = ""

        'the statistics function will be called with appropriate house
        'and database attribute order
        'colours change based on the house combo box value
        If cmbHouse.Text = "Green" Then
            Me.BackColor = &HC0FFC0
            Frame1.BackColor = &HC0FFC0
            cmdOk.BackColor = &HC0FFFF
            lblSnapCount.Caption = Statistics("Green", 1)
            lblCribbageCount.Caption = Statistics("Green", 2)
            lblSpillikinsCount.Caption = Statistics("Green", 3)
            lblScrabbleCount.Caption = Statistics("Green", 4)
            lblSnapWon.Caption = Statistics("Green", 5)
            lblCribbageWon.Caption = Statistics("Green", 6)
            lblSpillikinsWon.Caption = Statistics("Green", 7)
            lblScrabbleWon.Caption = Statistics("Green", 8)
            'call Tournamet won counting function
            lblTournaments.Caption = getTournaments("Green")
        ElseIf cmbHouse.Text = "Yellow" Then
            Me.BackColor = &HC0FFFF
            Frame1.BackColor = &HC0FFFF
            cmdOk.BackColor = &HC0FFC0
            lblSnapCount.Caption = Statistics("Yellow", 1)
            lblCribbageCount.Caption = Statistics("Yellow", 2)
            lblSpillikinsCount.Caption = Statistics("Yellow", 3)
            lblScrabbleCount.Caption = Statistics("Yellow", 4)
            lblSnapWon.Caption = Statistics("Yellow", 5)
            lblCribbageWon.Caption = Statistics("Yellow", 6)
            lblSpillikinsWon.Caption = Statistics("Yellow", 7)
            lblScrabbleWon.Caption = Statistics("Yellow", 8)
            'call Tournamet won counting function
            lblTournaments.Caption = getTournaments("Yellow")
        End If
    End If
End Sub

'*****************************************************************************
'a function that reads the number of tournaments won by strHouse from the database
'input: House name: strHouse
'return the number of tournaments won by strHouse
'*****************************************************************************
Private Function getTournaments(strHouse As String) As Integer
    gadoCommand.CommandText = "SELECT * FROM Tournaments" & _
        " WHERE House='" & strHouse & "'"
    Set gadoRecordSet = gadoCommand.Execute
    getTournaments = Val(gadoRecordSet.Fields(1))
End Function
```

```
frmHouseSt - 2

'*************************************************************************
'subroutine will be executed when the Ok command button is clicked
'*************************************************************************
Private Sub cmdOk_Click()
    frmMain.Show
    Unload Me
End Sub

'*************************************************************************
'subroutine will be executed when the form is loaded
'*************************************************************************
Private Sub Form_Load()
    cmbHouse.AddItem "Green"
    cmbHouse.AddItem "Yellow"
    'pre-select green
    cmbHouse.ListIndex = 0
    Me.BackColor = &HC0FFC0
End Sub

'*************************************************************************
'a function that calculate the sum of a specifi field in the database
'input: Class number: intClass
'input: the attribute location within the database (column position):intCol
'return the sum of the field
'*************************************************************************
Private Function Statistics(strHouse As String, intCol As Integer) As Integer
    Dim intX As Integer
    'integer value used to accumulate the values of the col
    Dim intCount As Integer

    'get the first record
    gadoCommand.CommandText = "SELECT * FROM Statistics WHERE House= '" _
    & strHouse & "'"
    Set gadoRecordSet = gadoCommand.Execute
     'intialise the accumulator
    intCount = Val(gadoRecordSet.Fields(intCol))

    'read the rest of the records
    For intX = 1 To 35
        gadoCommand.CommandText = "SELECT * FROM Statistics WHERE House= '" _
        & strHouse & "'"
        Set gadoRecordSet = gadoCommand.Execute
        gadoRecordSet.GetRows (intX)
        'if no more records belong to intClass exit the for loop
        If gadoRecordSet.EOF = True Then Exit For
        'add the current record field value to the accumulator
        intCount = intCount + Val(gadoRecordSet.Fields(intCol))
    Next intX
    'return the result
    Statistics = intCount
End Function
```

5.8 Classes' Achievement Statistics Form

5.8.1 User interface (Task4/a)

This form is loaded from the main window when the Classes' Achievements Statistics command button is clicked. It displays statistics about classes' achievement in the game. The information displayed about each Class are read and calculated from the database using SQL commands, as shown in the code below

```
'*************************************************************************
'a function that calculate the sum of a specifi field in the database
'input: Class number: intClass
'input: the attribute location within the database (column position):intCol
'return the sum of the field
'*************************************************************************
Private Function Statistics(intClass As Integer, intCol As Integer) As Integer
    Dim intX As Integer
    'integer value used to accumulate the values of the col
    Dim intCount As Integer
    'get the first record
    gadoCommand.CommandText = "SELECT * FROM Statistics WHERE Class= '" _
    & intClass & "'"
    Set gadoRecordSet = gadoCommand.Execute
    'intialise the accumulator
    intCount = Val(gadoRecordSet.Fields(intCol))
    'read the rest of the records
    For intX = 1 To 35
        gadoCommand.CommandText = "SELECT * FROM Statistics WHERE Class= '" _
        & intClass & "'"
        Set gadoRecordSet = gadoCommand.Execute
        gadoRecordSet.GetRows (intX)
        'if no more records belong to intClass exit the for loop
        If gadoRecordSet.EOF = True Then Exit For
        'add the current record field value to the accumulator
        intCount = intCount + Val(gadoRecordSet.Fields(intCol))
    Next intX
    'return the result
    Statistics = intCount
End Function
```

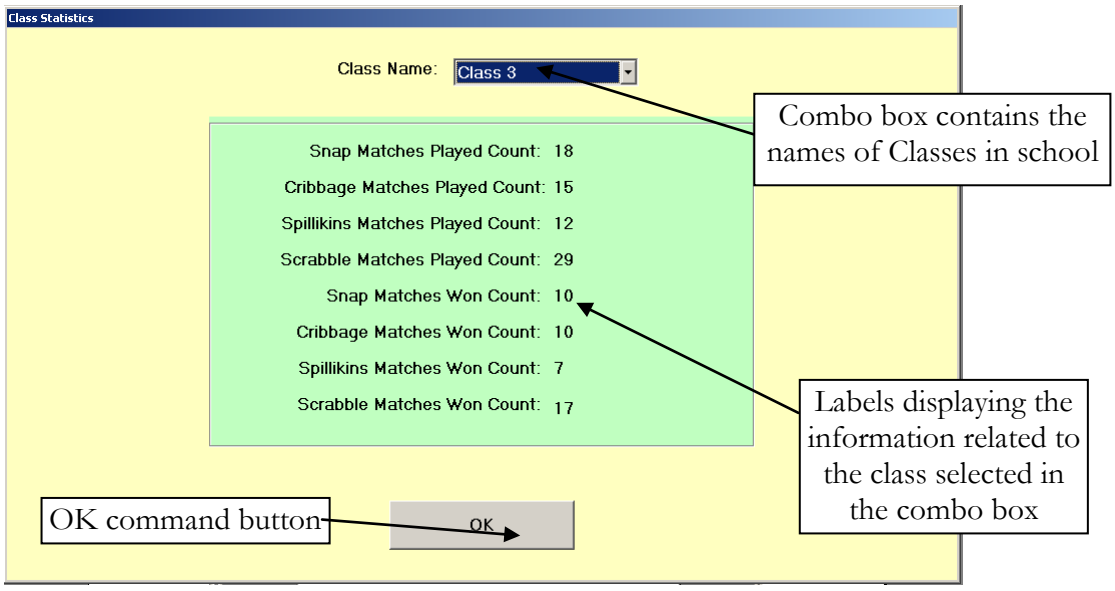


Figure 42

5.8.2 Testing (Task4/b)

To test each control, the correct data will be entered in all controls except the one that we are testing.

Test no.	Test	Test data	Expected Result	Actual Results	Comments
1	Ok Command Button	-	The main window will be loaded	The main window was loaded	Test passed
2	Class name combo box	Class1	The information about the class 1 will be displayed	The information about the class1 was displayed	Test passed
3	House name combo box	Class3	The information about the class 3 will be displayed	The information about the class 3 was displayed	Test passed

Table 21

I checked the results by referring to the access database and doing the calculations manually.

5.8.3 Code (frmClassSt.frm)

```
frmClassSt - 1

'************************** Tournament Organising System************************
'***********************************frmClassSt Code*****************************
'*****************************Programer: Somoud Saqfelhait**********************
'***********************************Date:07/06/2007*****************************
'*******************************************************************************
'this form will be loaded from the main menu form when Class Achievments
'Statistics  command button is clicked
Option Explicit

'*******************************************************************************
'subroutine will be executed when the class combo box is clicked to select item
'*******************************************************************************
Private Sub cmbClass_Click()
Dim intClass As Integer
    If cmbClass.ListIndex <> -1 Then
        'extract the class number
        intClass = Val(Right(cmbClass.Text, 1))
        'the statistics function will be called with appropriate class
        'and database attribute order
        lblSnapCount.Caption = Statistics(intClass, 1)
        lblCribbageCount.Caption = Statistics(intClass, 2)
        lblSpillikinsCount.Caption = Statistics(intClass, 3)
        lblScrabbleCount.Caption = Statistics(intClass, 4)
        lblSnapWon.Caption = Statistics(intClass, 5)
        lblCribbageWon.Caption = Statistics(intClass, 6)
        lblSpillikinsWon.Caption = Statistics(intClass, 7)
        lblScrabbleWon.Caption = Statistics(intClass, 8)

    End If
End Sub
```

```
'*************************************************************************
'subroutine will be executed when the Ok command button is clicked
'*************************************************************************
Private Sub cmdOk_Click()
    frmMain.Show
    Unload Me
End Sub

'*************************************************************************
'subroutine will be executed when the form is loaded
'*************************************************************************
Private Sub Form_Load()
    Dim intX As Integer
    'initialise the combo box
    For intX = 1 To 6
    cmbClass.AddItem "Class " & intX
    Next intX
    cmbClass.ListIndex = 0
    'Me.BackColor = &HC0FFC0
End Sub

'*************************************************************************
'a function that calculate the sum of a specifi field in the database
'input: Class number: intClass
'input: the attribute location within the database (column position):intCol
'return the sum of the field
'*************************************************************************
Private Function Statistics(intClass As Integer, intCol As Integer) As Integer
    Dim intX As Integer
    'integer value used to accumulate the values of the col
    Dim intCount As Integer
    'get the first record
    gadoCommand.CommandText = "SELECT * FROM Statistics WHERE Class= '" _
    & intClass & "'"
    Set gadoRecordSet = gadoCommand.Execute
    'intialise the accumulator
    intCount = Val(gadoRecordSet.Fields(intCol))

frmClassSt - 2


    'read the rest of the records
    For intX = 1 To 35
        gadoCommand.CommandText = "SELECT * FROM Statistics WHERE Class= '" _
        & intClass & "'"
        Set gadoRecordSet = gadoCommand.Execute
        gadoRecordSet.GetRows (intX)
        'if no more records belong to intClass exit the for loop
        If gadoRecordSet.EOF = True Then Exit For
        'add the current record field value to the accumulator
        intCount = intCount + Val(gadoRecordSet.Fields(intCol))
    Next intX
    'return the result
    Statistics = intCount
End Function
```

5.9 Create Games Form

5.9.1 User interface (Task4/a)

This form is loaded from the main window when the Games command button is clicked. It is used to select the number of matches to be played and the method of game schedule creation (manual or random). The original design had two forms to do the same operation; I found it more convenient to use one form only.

Validation and verification is used in coding to ensure that the numbers of matches entered are within the range (1-18). The sum of all the matches numbers entered should be odd to ensure that no draw can occur.

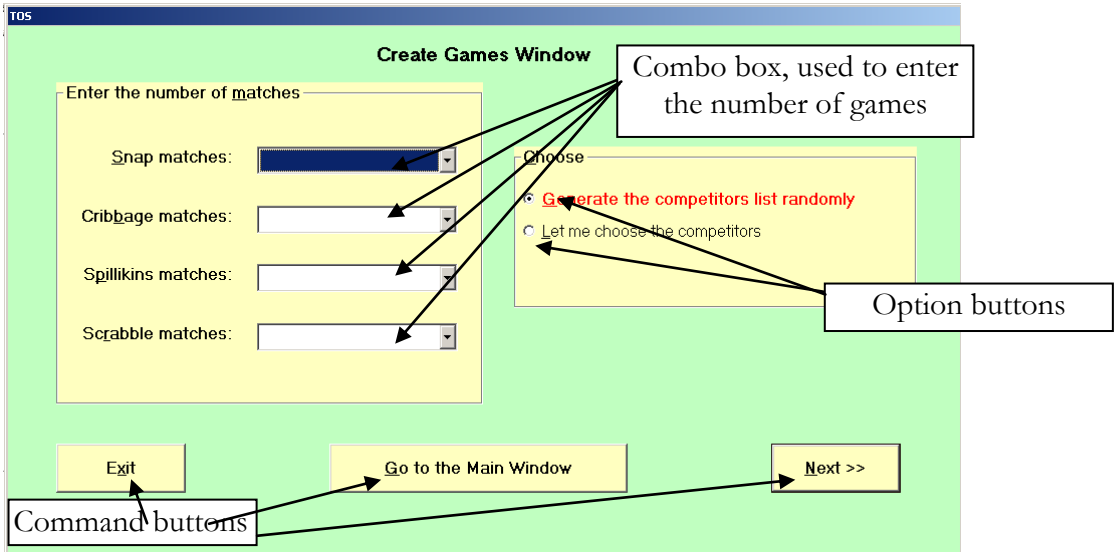


Figure 43

5.9.2 Testing (Task4/b)

To test each control, the correct data will be entered in all controls except the one that we are testing.

Test no.	Test	Test data	Expected Result	Actual Results	Comments
1	Exit Command Button	-	A message box will be displayed to ask the user to confirm if he/she want to exit the programme	A message box is displayed asking to confirm	Test passed
2	Go to main window button	-	The main window will be loaded	The main window was loaded	Test passed
3	Next command button + Generate randomly option button selected	-	The random game generation form will be loaded	The random game generation form is loaded	Test passed
4	Next command button + Generate manually option button	-	The manual game generation form will be loaded	The manual game generation form is loaded	Test passed
5	Next command button + Snap, cribbage, Spillikins and scrabble combo boxes	blank	a message will appear to ask the user to enter valid data 1-18	A message box (dialog) showed asking for valid data (1-18)	Test passed
6	Next command button + Snap, cribbage, Spillikins and scrabble combo boxes	A,2,1, blank	a message will appear to ask the user to enter valid data 1-18	A message box (dialog) showed asking for valid data (1-18)	Test passed
7	Next command button + Snap, cribbage, Spillikins and scrabble combo boxes	18,18, 18, 18	a message will appear to ask the user to make sure that the sum of all number of matches is odd	A message box (dialog) showed asking for the sum of all numbers to be odd	Test passed
8	Next command button + Snap, cribbage, Spillikins and scrabble combo boxes	18,18, 18, 17	If the random option button is selected the game random generation form will be loaded	With the random option button selected, the game random generation form was loaded	Test passed

Table 22

Evidence of this test is shown in the diagrams below

5.9.3 Testing Evidence (Task4/b)

Evidence of this test is shown in the diagrams below; the test number refers to the test number field in table 21

Test no. 1

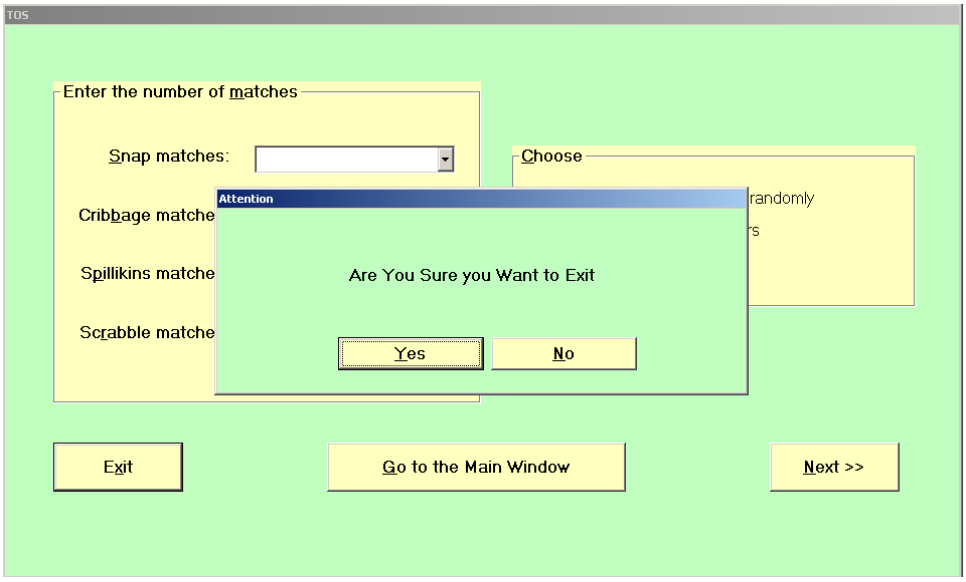


Figure 44

Test no. 5

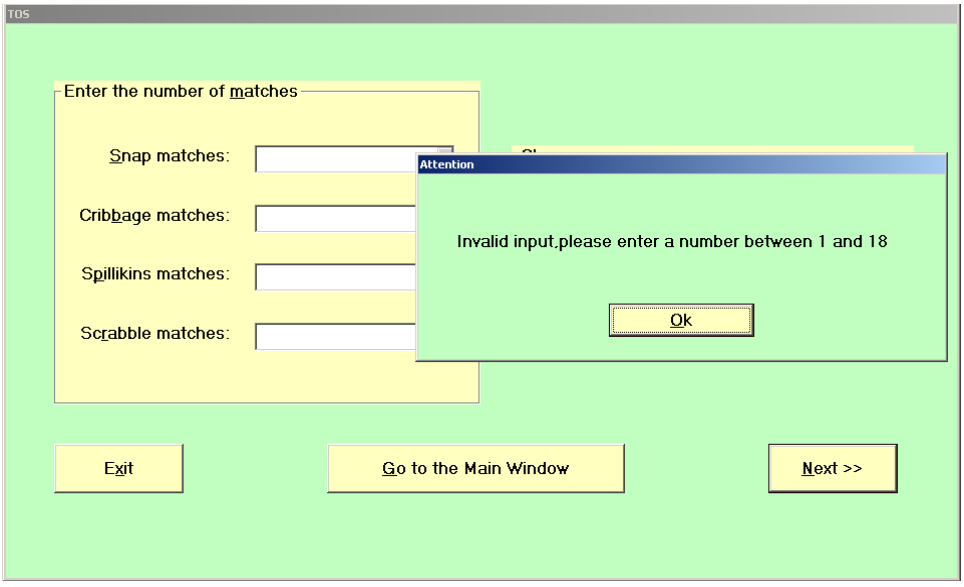


Figure 45

Test no. 7

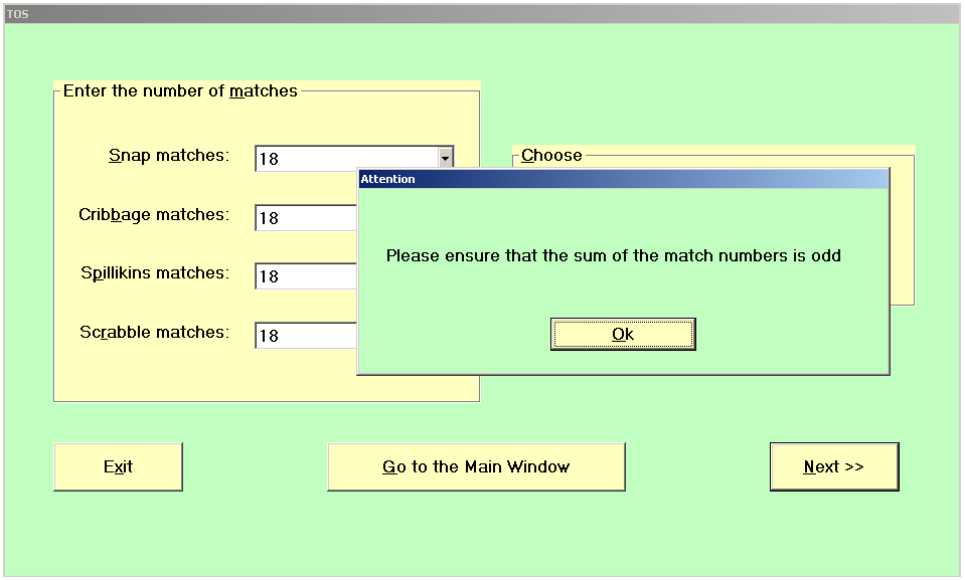


Figure 46

Test no. 8

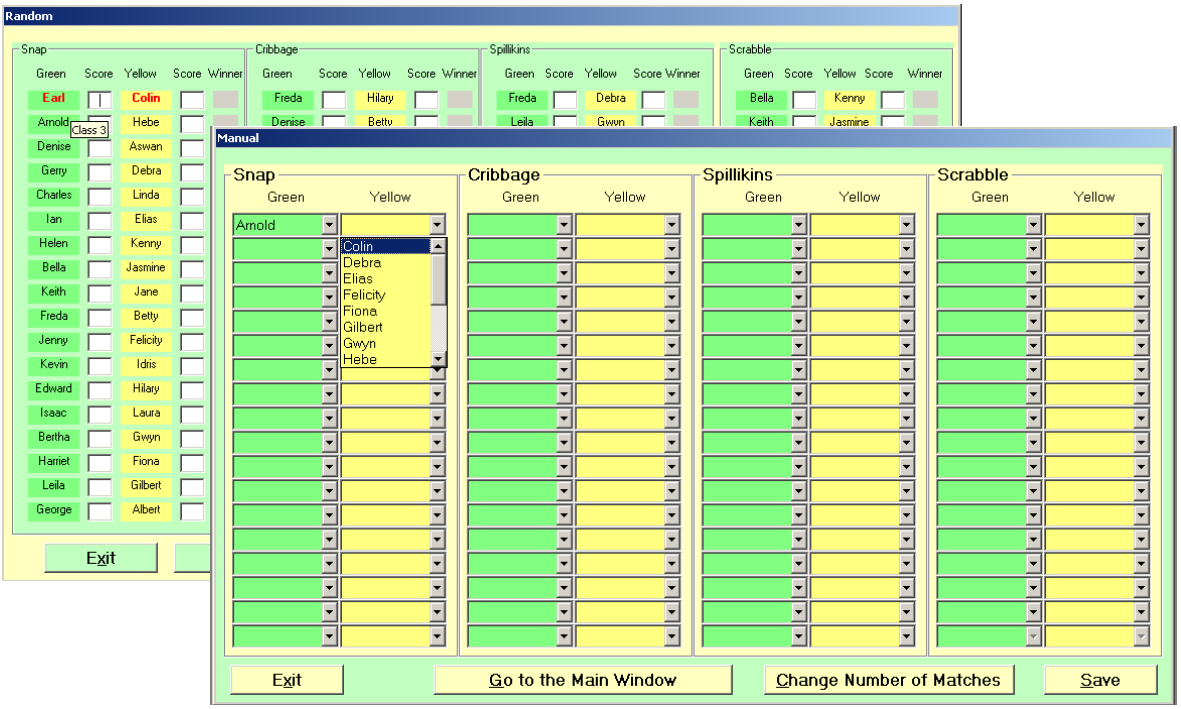


Figure 47

5.9.4 Code (frmIntro.frm) (Task4/a)

```
frmIntro - 1

'************************* Tournament Organising System***********************
'*********************************frmIntro Code*******************************
'****************************Programer: Somoud Saqfelhait***********************
'*********************************Date:07/04/2007*****************************
'*****************************************************************************
'this form will be loaded from the main menu form when Game button is clicked
Option Explicit

'*****************************************************************************
'subroutine will be executed when the "Go to Main window" button is clicked
'*****************************************************************************
Private Sub cmdMain_Click()
     frmMain.Show
    Unload Me
End Sub

'*****************************************************************************
'subroutine will be executed when the Exit command button is clicked
'*****************************************************************************
Private Sub cmdExit_Click()
  'define a variable to capture the msgbox response
    Dim varResponce As Variant
    varResponce = myMsgBox("Are You Sure you Want to Exit", "YesNo", "Attention")
    'exit program if response if Yes
    If varResponce = vbYes Then
        DB_Disconnect
        End
    End If
End Sub

'*****************************************************************************
'subroutine will be executed when the Next command button is clicked
'*****************************************************************************
Private Sub cmdNext_Click()
    'check for validity of number of matches intered
    If Not (IsNumeric(cmbSnap.Text) And IsNumeric(cmbCribbage.Text) _
        And IsNumeric(cmbSpillikins.Text) And IsNumeric(cmbScrabble.Text)) Then

            myMsgBox "Invalid input,please enter a number between 1 and 18" _
            , "Ok", "Attention"
            Exit Sub
    End If
    'check if entered number are in the range 1-18
    If Not ((1 <= Val(cmbSnap.Text) Or Val(cmbSnap.Text) <= 18) _
        And (1 <= Val(cmbCribbage.Text) And Val(cmbCribbage.Text) <= 18) _
        And (1 <= Val(cmbSpillikins.Text) And Val(cmbSpillikins.Text) <= 18) _
        And (1 <= Val(cmbScrabble.Text) And Val(cmbScrabble.Text) <= 18)) Then

            myMsgBox "Invalid number,please enter a number between 1 and 18" _
            , "Ok", "Attention"
            Exit Sub
    End If
    If (Val(cmbSnap.Text) + Val(cmbCribbage.Text) + _
        Val(cmbSpillikins.Text) + Val(cmbScrabble.Text)) Mod 2 = 0 Then
            myMsgBox "Please ensure that the sum of the match numbers is odd" _
            , "Ok", "Attention"
            Exit Sub
    End If
    'store the seleceted numbers of matches
    gintSnapMatch = Val(cmbSnap.Text)
    gintCribbageMatch = Val(cmbCribbage.Text)
    gintSpillikinsMatch = Val(cmbSpillikins.Text)
    gintScrabbleMatch = Val(cmbScrabble.Text)

    'load random or manual form based on the option button selected
    If gboolRandom = True Then
        frmRandom.Show
        'Me.Hide
        Unload Me
```

```
frmIntro - 2

    Else
        frmManual.Show
        'Me.Hide
        Unload Me
    End If

End Sub

'*************************************************************************
'this subroutine will be executed when the form is loaded
'*************************************************************************
Private Sub Form_Load()

    Dim intX As Integer
    'fix form position relative to the screen
    Left = Screen.Width / 2 - Width / 2
    Top = Screen.Height / 2 - Height / 2 - 2000
    'initialise the combo boxes
    For intX = 1 To 18
        cmbSnap.AddItem intX
        cmbCribbage.AddItem intX
        cmbSpillikins.AddItem intX
        cmbScrabble.AddItem intX
    Next intX
    gboolRandom = True
    gboolEdit = False
End Sub

'*************************************************************************
'this subroutine will be executed the mouse is moved over frame2
'*************************************************************************
Private Sub Frame2_MouseMove(Button As Integer, Shift As Integer, _
X As Single, Y As Single)

    optRandom.ForeColor = vbBlack
    optRandom.FontBold = False
    optManual.ForeColor = vbBlack
    optManual.FontBold = False
End Sub

'*************************************************************************
'this subroutine will be executed the mouse is moved over Manual option button
'*************************************************************************
Private Sub optManual_MouseMove(Button As Integer, Shift As Integer, _
X As Single, Y As Single)

    optManual.ForeColor = vbRed
    optManual.FontBold = True
End Sub

'*************************************************************************
'this subroutine will be executed the mouse is moved over Random option button
'*************************************************************************
Private Sub optRandom_MouseMove(Button As Integer, Shift As Integer, _
X As Single, Y As Single)

    optRandom.ForeColor = vbRed
    optRandom.FontBold = True
End Sub

'*************************************************************************
'this subroutine will be executed the Manual option button is clicked
'*************************************************************************
Private Sub optManual_Click()
    gboolRandom = False
End Sub

'*************************************************************************
'this subroutine will be executed the Random option button is clicked
'*************************************************************************
Private Sub optRandom_Click()
    gboolRandom = True
End Sub
```

5.10 Random Games Schedule Generation Form

5.10.1 User interface (Task4/a)

This form is loaded from the Create Game window when the next command button is clicked with the random option button selected. Or from the manual generation form when the save command button is clicked.

I have added the features below to verify that the games' rules are fulfilled.

- I have stored each student class in its label box "ToolTipText" property, so that when the mouse is moved over that label the class to which the student belongs is shown
- For the first game i.e. snap I added a subroutine connected to mouse move over student labels. The subroutine will change the font to bold/red and as well as the labels for the opponent in each other game. This way I was able to check if the student plays different opponent in each game.

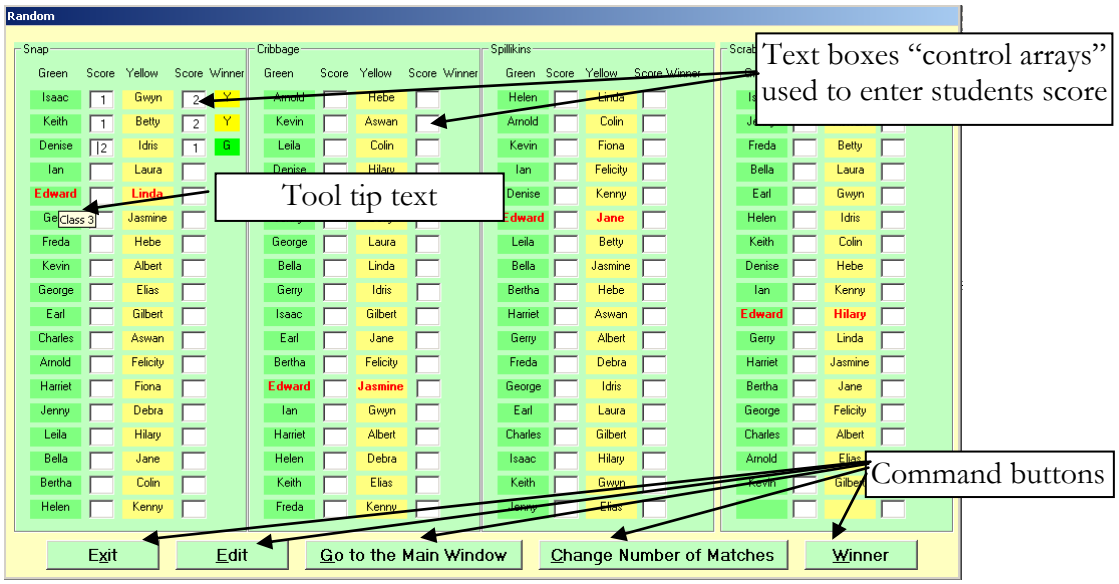


Figure 48

Other features are

- If winner command button is clicked and not all the results were entered the programme highlights the empty result boxes and ask the user to fill them.
- If Edit command button is clicked the manual generation form will be shown with its combo boxes showing the students. This allows the user to change some of the matches or all of them.
- Change Number of matches command button will load the Create game window.

5.10.2 Testing (Task4/b)

To test each control, the correct data will be entered in all controls except the one that we are testing.

Test no.	Test	Test data	Expected Result	Actual Results	Comments
1	Green house results (text boxes control array)	Blank	When "Winner" button is clicked the programme will highlight all the blank text boxes and ask the user to fill the empty highlighted boxes	When "Winner" was clicked the programme highlighted all the blank text boxes and asked for the empty highlighted boxes to be filled	Test passed
2	Green house results (text boxes control array)	a	When the text box lose focus a message box will appear and ask the user to enter valid result 0-3	When the text box lost focus a message box appeared and ask valid result 0-3	Test passed
3	Green house results (text boxes control array)	3	When the text box loses focus the corresponding text box "yellow results" will be updated with its result automatically.(0)	When the text box lost focus the corresponding text box "yellow results" was updated with its result automatically.(0)	Test passed
4	Green house results (text boxes control array)	4	When the text box lose focus a message box will appear and ask the user to enter valid result 0-3	When the text box lost focus a message box appeared and ask valid result 0-3	Test passed
5	Yellow house results (text boxes control array)	a	When the text box lose focus a message box will appear and ask the user to enter valid result 0-3	When the text box lost focus a message box appeared and ask valid result 0-3	Test passed
6	Yellow house results (text boxes control array)	3	When the text box loses focus the corresponding text box "green results" will be updated with its result automatically.(0)	When the text box lost focus the corresponding text box "green results" was updated with its result automatically.(0)	Test passed
7	Yellow house results (text boxes control array)	4	When the text box lose focus a message box will appear and ask the user to enter valid result 0-3	When the text box lost focus a message box appeared and ask valid result 0-3	Test passed
8	Exit Command Button	-	A message box will be displayed to ask the user to confirm if he/she want to exit the programme	A message box is displayed asking to confirm	Test passed
9	Go to main window button	-	The main window will be loaded	The main window was loaded	Test passed
10	Edit command button	-	The manual window will be loaded with the combo boxes displaying the students	The manual window was loaded with the combo boxes displaying the students	Test passed

Table 23

5.10.3 Testing Evidence (Task4/b)

Evidence of this test is shown in the diagrams below; the test number refers to the test number field in table 22

Test no. 1

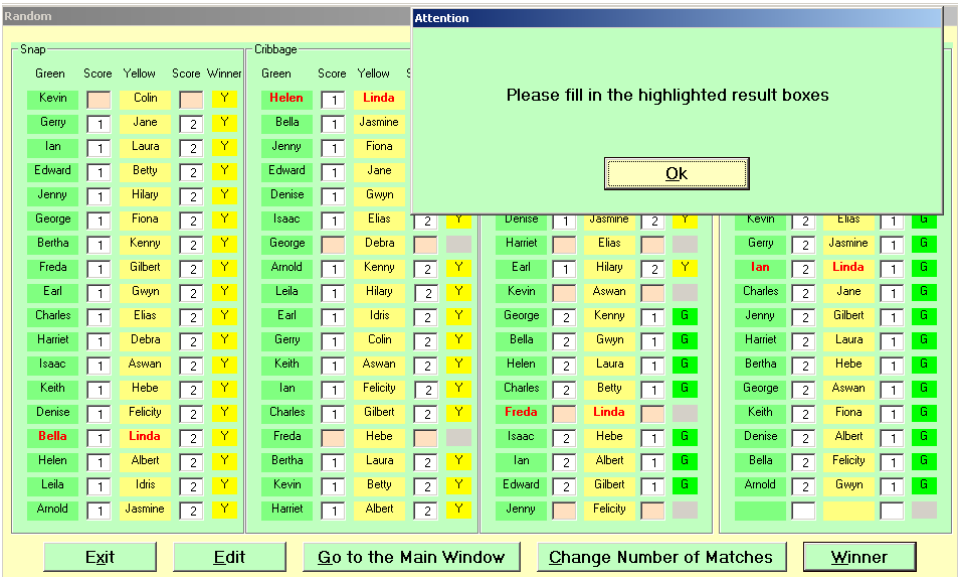


Figure 49

Test no. 2, 3, 4

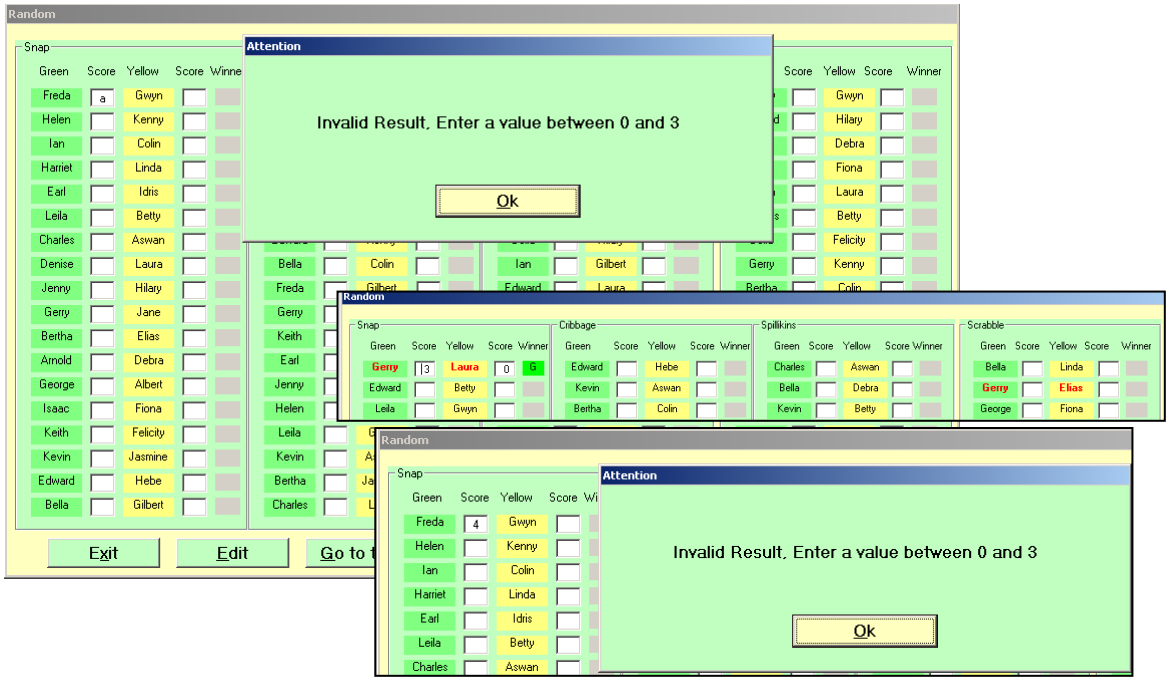


Figure 50

Test no. 10

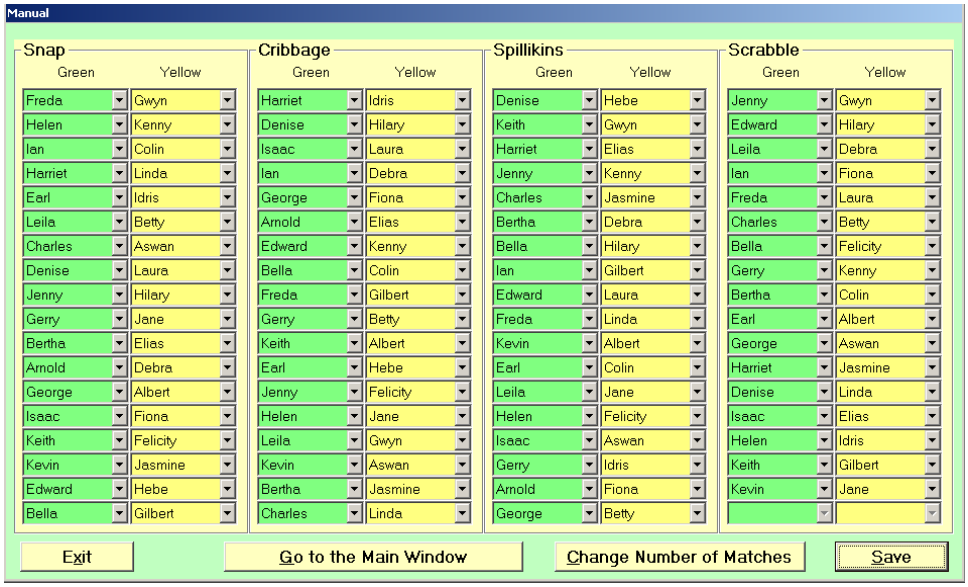


Figure 51

5.10.4 Code (frmRandom.frm) (Task4/a)

```
frmRandom - 1

'*************************************************************************************
'************************** Tournament Organising System**************************
'*********************************frmRandom Code*********************************
'*******************************Programer: S. Saqfelhait**************************
'*************************************Date:07/04/2007*****************************
'*************************************************************************************
'this form is loaded from frmIntro "select game generation method and number of
'matches form, it is also loaded from manual form when save button is clicked
Option Explicit

'*************************************************************************************
'subroutine will be executed when the Change command button is clicked
'*************************************************************************************
Private Sub cmdChange_Click()
    frmIntro.Show
    Unload Me
End Sub

'*************************************************************************************
'subroutine will be executed when the Edit command button is clicked
'*************************************************************************************
Private Sub cmdEdit_Click()
    gboolEdit = True
    frmManual.Show
    Unload Me
End Sub

'*************************************************************************************
'subroutine will be executed when the Exit command button is clicked
'*************************************************************************************
Private Sub cmdExit_Click()
    'define a variable to capture the msgbox response
    Dim varResponce As Variant
    varResponce = myMsgBox("Are You Sure you Want to Exit", _
                  "YesNo", "Attention")
    'exit program if response if Yes
    If varResponce = vbYes Then
        DB_Disconnect
        End
    End If
End Sub
```

```
frmRandom - 2

'*************************************************************************
'subroutine will be executed when the Go to main window command button is clicked
'*************************************************************************
Private Sub cmdMain_Click()
    frmMain.Show
    Unload Me
End Sub

'*************************************************************************
'subroutine will be executed when the Winner command button is clicked
'*************************************************************************
Private Sub cmdWinner_Click()
    'define a variable to count the green house points
    Dim GPoints As Integer
    'define a variable to count the yellow house points
    Dim YPoints As Integer
    Dim intX As Integer
    'define a variable to be used as a flag to track any empty result textbox
    Dim emptyResults As Boolean
    'initialise the variables
    GPoints = 0
    YPoints = 0
    emptyResults = False

    'find and highlight empty results text boxes
     For intX = 0 To gintSnapMatch - 1
        If txtGSnapResult(intX).Text = "" Then
            txtGSnapResult(intX).BackColor = &HC0E0FF
            txtYSnapResult(intX).BackColor = &HC0E0FF
            emptyResults = True
        Else
            txtGSnapResult(intX).BackColor = vbWhite
            txtYSnapResult(intX).BackColor = vbWhite
        End If
    Next intX

    For intX = 0 To gintCribbageMatch - 1
        If txtGCribbageResult(intX).Text = "" Then
            txtGCribbageResult(intX).BackColor = &HC0E0FF
            txtYCribbageResult(intX).BackColor = &HC0E0FF
            emptyResults = True
        Else
            txtGCribbageResult(intX).BackColor = vbWhite
            txtYCribbageResult(intX).BackColor = vbWhite
        End If
    Next intX

    For intX = 0 To gintSpillikinsMatch - 1
        If txtGSpillikinsResult(intX).Text = "" Then
            txtGSpillikinsResult(intX).BackColor = &HC0E0FF
            txtYSpillikinsResult(intX).BackColor = &HC0E0FF
            emptyResults = True
        Else
            txtGSpillikinsResult(intX).BackColor = vbWhite
            txtYSpillikinsResult(intX).BackColor = vbWhite
        End If

    Next intX

    For intX = 0 To gintScrabbleMatch - 1
        If txtGScrabbleResult(intX).Text = "" Then
            txtGScrabbleResult(intX).BackColor = &HC0E0FF
            txtYScrabbleResult(intX).BackColor = &HC0E0FF
            emptyResults = True
        Else
            txtGScrabbleResult(intX).BackColor = vbWhite
            txtYScrabbleResult(intX).BackColor = vbWhite
        End If
    Next intX

    'if any empty result text boxes were found tell the user and exit the sub
    If emptyResults = True Then
        myMsgBox "Please fill in the highlighted result boxes" _
        , "Ok", "Attention" '
        Exit Sub
```

```
frmRandom - 2

'*************************************************************************
'subroutine will be executed when the Go to main window command button is clicke
'*************************************************************************
Private Sub cmdMain_Click()
    frmMain.Show
    Unload Me
End Sub

'*************************************************************************
'subroutine will be executed when the Winner command button is clicked
'*************************************************************************
Private Sub cmdWinner_Click()
    'define a variable to count the green house points
    Dim GPoints As Integer
    'define a variable to count the yellow house points
    Dim YPoints As Integer
    Dim intX As Integer
    'define a variable to be used as a flag to track any empty result textbox
    Dim emptyResults As Boolean
    'initialise the variables
    GPoints = 0
    YPoints = 0
    emptyResults = False

    'find and highlight empty results text boxes
     For intX = 0 To gintSnapMatch - 1
        If txtGSnapResult(intX).Text = "" Then
            txtGSnapResult(intX).BackColor = &HC0E0FF
            txtYSnapResult(intX).BackColor = &HC0E0FF
            emptyResults = True
        Else
            txtGSnapResult(intX).BackColor = vbWhite
            txtYSnapResult(intX).BackColor = vbWhite
        End If
    Next intX

    For intX = 0 To gintCribbageMatch - 1
        If txtGCribbageResult(intX).Text = "" Then
            txtGCribbageResult(intX).BackColor = &HC0E0FF
            txtYCribbageResult(intX).BackColor = &HC0E0FF
            emptyResults = True
        Else
            txtGCribbageResult(intX).BackColor = vbWhite
            txtYCribbageResult(intX).BackColor = vbWhite
        End If
    Next intX

    For intX = 0 To gintSpillikinsMatch - 1
        If txtGSpillikinsResult(intX).Text = "" Then
            txtGSpillikinsResult(intX).BackColor = &HC0E0FF
            txtYSpillikinsResult(intX).BackColor = &HC0E0FF
            emptyResults = True
        Else
            txtGSpillikinsResult(intX).BackColor = vbWhite
            txtYSpillikinsResult(intX).BackColor = vbWhite
        End If

    Next intX

    For intX = 0 To gintScrabbleMatch - 1
        If txtGScrabbleResult(intX).Text = "" Then
            txtGScrabbleResult(intX).BackColor = &HC0E0FF
            txtYScrabbleResult(intX).BackColor = &HC0E0FF
            emptyResults = True
        Else
            txtGScrabbleResult(intX).BackColor = vbWhite
            txtYScrabbleResult(intX).BackColor = vbWhite
        End If
    Next intX

    'if any empty result text boxes were found tell the user and exit the sub
    If emptyResults = True Then
        myMsgBox "Please fill in the highlighted result boxes" _
        , "Ok", "Attention" '
        Exit Sub
```

```
frmRandom - 3

    End If
    'calculate the results
    For intX = 0 To gintSnapMatch - 1
        If Val(txtGSnapResult(intX).Text) > _
        Val(txtYSnapResult(intX).Text) Then
            Call UpdateMatchWonCount(strGame(intX, 0), "Snap", _
                Val(txtGSnapResult(intX).Text))
            GPoints = GPoints + 1
        Else
            Call UpdateMatchWonCount(strGame(intX, 1), "Snap", _
            Val(txtYSnapResult(intX).Text))
            YPoints = YPoints + 1
        End If
    Next intX

    For intX = 0 To gintCribbageMatch - 1
        If Val(txtGCribbageResult(intX).Text) > _
        Val(txtYCribbageResult(intX).Text) Then
            Call UpdateMatchWonCount(strGame(intX, 2), "Cribbage" _
                , Val(txtGCribbageResult(intX).Text))
            GPoints = GPoints + 1
        Else
            Call UpdateMatchWonCount(strGame(intX, 3), "Cribbage" _
                , Val(txtYCribbageResult(intX).Text))
            YPoints = YPoints + 1
        End If
    Next intX

    For intX = 0 To gintSpillikinsMatch - 1
        If Val(txtGSpillikinsResult(intX).Text) > _
        Val(txtYSpillikinsResult(intX).Text) Then
            Call UpdateMatchWonCount(strGame(intX, 4), "Spillikins" _
                , Val(txtGSpillikinsResult(intX).Text))
            GPoints = GPoints + 1
        Else
            Call UpdateMatchWonCount(strGame(intX, 5), "Spillikins" _
                , Val(txtYSpillikinsResult(intX).Text))
            YPoints = YPoints + 1
        End If
    Next intX

    For intX = 0 To gintScrabbleMatch - 1
        If Val(txtGScrabbleResult(intX).Text) > _
        Val(txtYScrabbleResult(intX).Text) Then
            Call UpdateMatchWonCount(strGame(intX, 6), "Scrabble" _
                , Val(txtGScrabbleResult(intX).Text))
            GPoints = GPoints + 1
        Else
            Call UpdateMatchWonCount(strGame(intX, 7), "Scrabble" _
                , Val(txtGScrabbleResult(intX).Text))
            YPoints = YPoints + 1
        End If
    Next intX

    'determine the winner
    If GPoints > YPoints Then
        gstrWinner = "Green"
    Else
        gstrWinner = "Yellow"
    End If
    Dim tempCount As Integer
    'store game results and players in the database
    For intX = 0 To gintSnapMatch - 1
        Call UpdateMatchCount(strGame(intX, 0), "Snap")
        Call UpdateMatchCount(strGame(intX, 1), "Snap")
    Next intX

    For intX = 0 To gintCribbageMatch - 1
        Call UpdateMatchCount(strGame(intX, 2), "Cribbage")
        Call UpdateMatchCount(strGame(intX, 3), "Cribbage")
    Next intX

    For intX = 0 To gintSpillikinsMatch - 1
        Call UpdateMatchCount(strGame(intX, 4), "Spillikins")
        Call UpdateMatchCount(strGame(intX, 5), "Spillikins")
```

```
frmRandom - 4

    Next intX

    For intX = 0 To gintScrabbleMatch - 1
        Call UpdateMatchCount(strGame(intX, 6), "Scrabble")
        Call UpdateMatchCount(strGame(intX, 7), "Scrabble")
    Next intX
    Call UpdateTournaments(gstrWinner)

    Unload Me
    frmWinner.Show
End Sub

'*************************************************************************
'subroutine will be executed when the form is activated
'*************************************************************************
Private Sub Form_Activate()
    'if the form was loaded from Manual form i.e gboolrandom=false
    If gboolRandom = False Then
        Call displayManual
    End If
    'find the class of each student and store it in each label box tiptool
    Dim intX As Integer
    'snap
    For intX = 0 To gintSnapMatch - 1
    'green opponent
    gadoCommand.CommandText = "SELECT * FROM Students WHERE studentName = '" & _
        lblGSnapStudent(intX).Caption & "'"
                Set gadoRecordSet = gadoCommand.Execute
        lblGSnapStudent(intX).ToolTipText = "Class " & gadoRecordSet.Fields(2)
     'yellow opponent
        gadoCommand.CommandText = "SELECT * FROM Students WHERE studentName = '" & _
        lblYSnapStudent(intX).Caption & "'"
                Set gadoRecordSet = gadoCommand.Execute
        lblYSnapStudent(intX).ToolTipText = "Class " & gadoRecordSet.Fields(2)
     Next intX
     'cribbage
     For intX = 0 To gintCribbageMatch - 1
        'green
        gadoCommand.CommandText = "SELECT * FROM Students WHERE studentName = '" & _
        lblGCribbageStudent(intX).Caption & "'"
                Set gadoRecordSet = gadoCommand.Execute
        lblGCribbageStudent(intX).ToolTipText = "Class " & gadoRecordSet.Fields(2)

         'yellow
         gadoCommand.CommandText = "SELECT * FROM Students WHERE studentName = '" & _
        lblYCribbageStudent(intX).Caption & "'"
                Set gadoRecordSet = gadoCommand.Execute
        lblYCribbageStudent(intX).ToolTipText = "Class " & gadoRecordSet.Fields(2)
    Next intX
    'spillikins
    For intX = 0 To gintSpillikinsMatch - 1
        'green
        gadoCommand.CommandText = "SELECT * FROM Students WHERE studentName = '" & _
        lblGSpillikinsStudent(intX).Caption & "'"
                Set gadoRecordSet = gadoCommand.Execute
        lblGSpillikinsStudent(intX).ToolTipText = "Class " & gadoRecordSet.Fields(2)

         'yellow
         gadoCommand.CommandText = "SELECT * FROM Students WHERE studentName = '" & _
        lblYSpillikinsStudent(intX).Caption & "'"
                Set gadoRecordSet = gadoCommand.Execute
        lblYSpillikinsStudent(intX).ToolTipText = "Class " & gadoRecordSet.Fields(2)
    Next intX
    'scrabble
    For intX = 0 To gintScrabbleMatch - 1
    'green
        gadoCommand.CommandText = "SELECT * FROM Students WHERE studentName = '" & _
        lblGScrabbleStudent(intX).Caption & "'"
                Set gadoRecordSet = gadoCommand.Execute
        lblGScrabbleStudent(intX).ToolTipText = "Class " & gadoRecordSet.Fields(2)

         'yellow
         gadoCommand.CommandText = "SELECT * FROM Students WHERE studentName = '" & _
        lblYScrabbleStudent(intX).Caption & "'"
                Set gadoRecordSet = gadoCommand.Execute
```

```
frmRandom - 5

        lblYScrabbleStudent(intX).ToolTipText = "Class " & gadoRecordSet.Fields(2)

    Next intX
End Sub

'*************************************************************************
'subroutine will be executed when the from is loaded
'*************************************************************************
Private Sub Form_Load()

    Dim intX As Integer
    'disable result boxes corresponding to no opponents
    For intX = gintSnapMatch To 17
        txtGSnapResult(intX).Enabled = False
        txtYSnapResult(intX).Enabled = False
    Next intX

    For intX = gintCribbageMatch To 17
        txtGCribbageResult(intX).Enabled = False
        txtYCribbageResult(intX).Enabled = False
    Next intX

    For intX = gintSpillikinsMatch To 17
        txtGSpillikinsResult(intX).Enabled = False
        txtYSpillikinsResult(intX).Enabled = False
    Next intX

    For intX = gintScrabbleMatch To 17
        txtGScrabbleResult(intX).Enabled = False
        txtYScrabbleResult(intX).Enabled = False
    Next intX

    'if the form was loaded from Manual form i.e gboolrandom=false
    'display the players selected in the manual form
    If gboolRandom = False Then
        Call displayManual
    Else
    'generate a game schedule randomly
        Call GenerateRandom
    End If
End Sub



'*************************************************************************
'subroutine will be executed when the mouse is moved on label number "Index" green
'*************************************************************************
Private Sub lblGSnapStudent_MouseMove(Index As Integer, Button As Integer, _
Shift As Integer, X As Single, Y As Single)

    Dim intX As Integer

    For intX = 0 To gintSnapMatch - 1
        If intX = Index Then
            lblGSnapStudent(intX).ForeColor = vbRed
            lblYSnapStudent(intX).ForeColor = vbRed
            lblGSnapStudent(intX).FontBold = True
            lblYSnapStudent(intX).FontBold = True
        Else
            lblGSnapStudent(intX).ForeColor = vbBlack
            lblYSnapStudent(intX).ForeColor = vbBlack
            lblGSnapStudent(intX).FontBold = False
            lblYSnapStudent(intX).FontBold = False
        End If
    Next intX
    'highlight corresponding labels from the rest of the games and
    'highlight opponents
    For intX = 0 To gintCribbageMatch - 1
        If lblGCribbageStudent(intX).Caption = lblGSnapStudent(Index).Caption Then
            lblGCribbageStudent(intX).ForeColor = vbRed
            lblYCribbageStudent(intX).ForeColor = vbRed
            lblGCribbageStudent(intX).FontBold = True
            lblYCribbageStudent(intX).FontBold = True
        Else
            lblGCribbageStudent(intX).ForeColor = vbBlack
```

```
frmRandom - 6

            lblYCribbageStudent(intX).ForeColor = vbBlack
            lblGCribbageStudent(intX).FontBold = False
            lblYCribbageStudent(intX).FontBold = False
        End If
    Next intX

     For intX = 0 To gintSpillikinsMatch - 1
        If lblGSpillikinsStudent(intX).Caption = lblGSnapStudent(Index).Caption Then
            lblGSpillikinsStudent(intX).ForeColor = vbRed
            lblYSpillikinsStudent(intX).ForeColor = vbRed
            lblGSpillikinsStudent(intX).FontBold = True
            lblYSpillikinsStudent(intX).FontBold = True
        Else
            lblGSpillikinsStudent(intX).ForeColor = vbBlack
            lblYSpillikinsStudent(intX).ForeColor = vbBlack
            lblGSpillikinsStudent(intX).FontBold = False
            lblYSpillikinsStudent(intX).FontBold = False
        End If
    Next intX

    For intX = 0 To gintScrabbleMatch - 1
        If lblGScrabbleStudent(intX).Caption = lblGSnapStudent(Index).Caption Then
            lblGScrabbleStudent(intX).ForeColor = vbRed
            lblYScrabbleStudent(intX).ForeColor = vbRed
            lblGScrabbleStudent(intX).FontBold = True
            lblYScrabbleStudent(intX).FontBold = True
        Else
            lblGScrabbleStudent(intX).ForeColor = vbBlack
            lblYScrabbleStudent(intX).ForeColor = vbBlack
            lblGScrabbleStudent(intX).FontBold = False
            lblYScrabbleStudent(intX).FontBold = False
        End If
    Next intX

End Sub

'*************************************************************************
'subroutine will be executed when the mouse is moved on label number "Index" yellow
'*************************************************************************
Private Sub lblYSnapStudent_MouseMove(Index As Integer, Button As Integer, _
Shift As Integer, X As Single, Y As Single)

    Dim intX As Integer

    For intX = 0 To gintSnapMatch - 1
        If intX = Index Then
            lblGSnapStudent(intX).ForeColor = vbRed
            lblYSnapStudent(intX).ForeColor = vbRed
            lblGSnapStudent(intX).FontBold = True
            lblYSnapStudent(intX).FontBold = True
        Else
            lblGSnapStudent(intX).ForeColor = vbBlack
            lblYSnapStudent(intX).ForeColor = vbBlack
            lblGSnapStudent(intX).FontBold = False
            lblYSnapStudent(intX).FontBold = False
        End If
    Next intX
    'highlight corresponding labels from the rest of the games and
    'highlight opponents
    For intX = 0 To gintCribbageMatch - 1
        If lblYCribbageStudent(intX).Caption = lblYSnapStudent(Index).Caption Then
            lblGCribbageStudent(intX).ForeColor = vbRed
            lblYCribbageStudent(intX).ForeColor = vbRed
            lblGCribbageStudent(intX).FontBold = True
            lblYCribbageStudent(intX).FontBold = True
        Else
            lblGCribbageStudent(intX).ForeColor = vbBlack
            lblYCribbageStudent(intX).ForeColor = vbBlack
            lblGCribbageStudent(intX).FontBold = False
            lblYCribbageStudent(intX).FontBold = False
        End If
    Next intX

     For intX = 0 To gintSpillikinsMatch - 1
        If lblYSpillikinsStudent(intX).Caption = lblYSnapStudent(Index).Caption Then
```

frmRandom - 7

```
            lblGSpillikinsStudent(intX).ForeColor = vbRed
            lblYSpillikinsStudent(intX).ForeColor = vbRed
            lblGSpillikinsStudent(intX).FontBold = True
            lblYSpillikinsStudent(intX).FontBold = True
        Else
            lblGSpillikinsStudent(intX).ForeColor = vbBlack
            lblYSpillikinsStudent(intX).ForeColor = vbBlack
            lblGSpillikinsStudent(intX).FontBold = False
            lblYSpillikinsStudent(intX).FontBold = False
        End If
    Next intX

    For intX = 0 To gintScrabbleMatch - 1
        If lblYScrabbleStudent(intX).Caption = lblYSnapStudent(Index).Caption Then
            lblGScrabbleStudent(intX).ForeColor = vbRed
            lblYScrabbleStudent(intX).ForeColor = vbRed
            lblGScrabbleStudent(intX).FontBold = True
            lblYScrabbleStudent(intX).FontBold = True
        Else
            lblGScrabbleStudent(intX).ForeColor = vbBlack
            lblYScrabbleStudent(intX).ForeColor = vbBlack
            lblGScrabbleStudent(intX).FontBold = False
            lblYScrabbleStudent(intX).FontBold = False
        End If
    Next intX

End Sub

'*************************************************************************
'subroutine will be executed when the text is changed in any text box of the
'Green students Snap results text boxes control array
'*************************************************************************
Private Sub txtGSnapResult_Change(Index As Integer)
    'if the change results empty box, ignore the change
    If txtGSnapResult(Index) = "" Then Exit Sub

    'check the validity of input text
    If Val(txtGSnapResult(Index).Text) < 0 Or Val(txtGSnapResult(Index).Text) > 3 Or _
        Not IsNumeric((txtGSnapResult(Index).Text)) Then
        myMsgBox "Invalid Result, Enter a value between 0 and 3" _
        , "Ok", "Attention"
        txtGSnapResult(Index) = ""
        txtYSnapResult(Index) = ""
        txtGSnapResult(Index).SetFocus
        Exit Sub
    End If
    'compute the score of the opponent based on the value entered
    txtYSnapResult(Index).Text = _
        str$(3 - Val(txtGSnapResult(Index).Text))

    'determine the winner of the match
    If Val(txtGSnapResult(Index).Text) >= 2 Then
        lblSnapWinner(Index).BackColor = vbGreen
        lblSnapWinner(Index).Caption = "G"
    End If
End Sub

'*************************************************************************
'subroutine will be executed when the text is changed in any text box of the
'yellow students Snap results text boxes control array
'*************************************************************************
Private Sub txtYSnapResult_Change(Index As Integer)
    'if the change results empty box, ignore the change
    If txtYSnapResult(Index) = "" Then Exit Sub

    'check the validity of input text
    If Val(txtYSnapResult(Index).Text) < 0 Or Val(txtYSnapResult(Index).Text) > 3 Or _
    Not IsNumeric(txtYSnapResult(Index).Text) Then
        myMsgBox "Invalid Result, Enter a value between 0 and 3" _
        , "Ok", "Attention"
        txtYSnapResult(Index) = ""
        txtGSnapResult(Index) = ""
        txtYSnapResult(Index).SetFocus
        Exit Sub
    End If
```

```
frmRandom - 8

    'compute the score of the opponent based on the value entered
    txtGSnapResult(Index).Text = _
        str$(3 - Val(txtYSnapResult(Index).Text))

    'determine the winner of the match
    If Val(txtYSnapResult(Index).Text) >= 2 Then
        lblSnapWinner(Index).BackColor = vbYellow
        lblSnapWinner(Index).Caption = "Y"
    End If
End Sub

'*************************************************************************
'subroutine will be executed when the text is changed in any text box of the
'yellow students Snap results text boxes control array
'*************************************************************************
Private Sub txtGcribbageResult_Change(Index As Integer)
    'if the change results empty box, ignore the change
    If txtGCribbageResult(Index) = "" Then Exit Sub
    txtYCribbageResult(Index) = ""

    'check the validity of input text
    If Val(txtGCribbageResult(Index).Text) < 0 Or Val(txtGCribbageResult(Index).Text) > 3 Then
        myMsgBox "Invalid Result, Enter a value between 0 and 3" _
        , "Ok", "Attention"
        txtGCribbageResult(Index) = ""
        txtGCribbageResult(Index).SetFocus
        Exit Sub
    End If
    'compute the score of the opponent based on the value entered
    txtYCribbageResult(Index).Text = _
    str$(3 - Val(txtGCribbageResult(Index).Text))

    'determine the winner of the match
    If Val(txtGCribbageResult(Index).Text) >= 2 Then
        lblCribbageWinner(Index).BackColor = vbGreen
        lblCribbageWinner(Index).Caption = "G"
    End If

End Sub

'*************************************************************************
'subroutine will be executed when the text is changed in any text box of the
'yellow students Cribbage results text boxes control array
'*************************************************************************
Private Sub txtYCribbageResult_Change(Index As Integer)
    'if the change results empty box, ignore the change
    If txtYCribbageResult(Index) = "" Then Exit Sub

    'check the validity of input text
    If Val(txtYCribbageResult(Index).Text) < 0 Or Val(txtYCribbageResult(Index).Text) > 3 Then
        myMsgBox "Invalid Result, Enter a value between 0 and 3" _
        , "Ok", "Attention"
        txtYCribbageResult(Index) = ""
        txtGCribbageResult(Index) = ""
        txtYCribbageResult(Index).SetFocus
        Exit Sub
    End If

    'compute the score of the opponent based on the value entered
    txtGCribbageResult(Index).Text = _
        str$(3 - Val(txtYCribbageResult(Index).Text))

    'determine the winner of the match
    If Val(txtYCribbageResult(Index).Text) >= 2 Then
            lblCribbageWinner(Index).BackColor = vbYellow
            lblCribbageWinner(Index).Caption = "Y"
    End If
End Sub

'*************************************************************************
'subroutine will be executed when the text is changed in any text box of the
' Green students Spillikins results text boxes control array
'*************************************************************************
Private Sub txtGspillikinsResult_Change(Index As Integer)
    'if the change results empty box, ignore the change
```

frmRandom - 9

```
    If txtGSpillikinsResult(Index) = "" Then Exit Sub

    'check the validity of input text
    If Val(txtGSpillikinsResult(Index).Text) < 0 Or Val(txtGSpillikinsResult(Index).Text) > 3 Then
        myMsgBox "Invalid Result, Enter a value between 0 and 3" _
        , "Ok", "Attention"
        txtGSpillikinsResult(Index) = ""
        txtYSpillikinsResult(Index) = ""
        txtGSpillikinsResult(Index).SetFocus
        Exit Sub
    End If

    'compute the score of the opponent based on the value entered
    txtYSpillikinsResult(Index).Text = _
        str$(3 - Val(txtGSpillikinsResult(Index).Text))

    'determine the winner of the match
    If Val(txtGSpillikinsResult(Index).Text) >= 2 Then
        lblSpillikinsWinner(Index).BackColor = vbGreen
        lblSpillikinsWinner(Index).Caption = "G"
    End If
End Sub

'*************************************************************************
'subroutine will be executed when the text is changed in any text box of the
'yellow students Spillikins results text boxes control array
'*************************************************************************
Private Sub txtYSpillikinsResult_Change(Index As Integer)
    'if the change results empty box, ignore the change
    If txtYSpillikinsResult(Index) = "" Then Exit Sub

    'check the validity of input text
    If Val(txtYSpillikinsResult(Index).Text) < 0 Or Val(txtYSpillikinsResult(Index).Text) > 3 Then
        myMsgBox "Invalid Result, Enter a value between 0 and 3" _
        , "Ok", "Attention"
        txtYSpillikinsResult(Index) = ""
        txtGSpillikinsResult(Index) = ""
        txtYSpillikinsResult(Index).SetFocus
        Exit Sub
    End If

    'compute the score of the opponent based on the value entered
    txtGSpillikinsResult(Index).Text = _
        str$(3 - Val(txtYSpillikinsResult(Index).Text))

    'determine the winner of the match
     If Val(txtYSpillikinsResult(Index).Text) >= 2 Then
        lblSpillikinsWinner(Index).BackColor = vbYellow
        lblSpillikinsWinner(Index).Caption = "Y"
    End If
End Sub

'*************************************************************************
'subroutine will be executed when the text is changed in any text box of the
'green students Scrabble results text boxes control array
'*************************************************************************
Private Sub txtGScrabbleResult_Change(Index As Integer)
    'if the change results empty box, ignore the change
    If txtGScrabbleResult(Index) = "" Then Exit Sub

    'check the validity of input text
    If Val(txtGScrabbleResult(Index).Text) < 0 Or Val(txtGScrabbleResult(Index).Text) > 3 Then
        myMsgBox "Invalid Result, Enter a value between 0 and 3" _
        , "Ok", "Attention"
        txtGScrabbleResult(Index) = ""
        txtYScrabbleResult(Index) = ""
        txtGScrabbleResult(Index).SetFocus
        Exit Sub
    End If

    'compute the score of the opponent based on the value entered
    txtYScrabbleResult(Index).Text = _
        str$(3 - Val(txtGScrabbleResult(Index).Text))

    'determine the winner of the match
```

```
frmRandom - 10

    If Val(txtGScrabbleResult(Index).Text) >= 2 Then
        lblScrabbleWinner(Index).BackColor = vbGreen
        lblScrabbleWinner(Index).Caption = "G"
    End If
End Sub

'*************************************************************************
'subroutine will be executed when the text is changed in any text box of the
'yellow students Scrabble results text boxes control array
'*************************************************************************
Private Sub txtYscrabbleResult_Change(Index As Integer)
    'if the change results empty box, ignore the change
    If txtYScrabbleResult(Index) = "" Then Exit Sub

    'check the validity of input text
    If Val(txtYScrabbleResult(Index).Text) < 0 Or Val(txtYScrabbleResult(Index).Text) > 3 Then
        myMsgBox "Invalid Result, Enter a value between 0 and 3" _
        , "Ok", "Attention"
        txtYScrabbleResult(Index) = ""
        txtGScrabbleResult(Index) = ""
        txtYScrabbleResult(Index).SetFocus
        Exit Sub
    End If

    'compute the score of the opponent based on the value entered
    txtGScrabbleResult(Index).Text = _
        str$(3 - Val(txtYScrabbleResult(Index).Text))

    'determine the winner of the match
    If Val(txtYScrabbleResult(Index).Text) >= 2 Then
        lblScrabbleWinner(Index).BackColor = vbYellow
        lblScrabbleWinner(Index).Caption = "Y"
    End If
End Sub

'*************************************************************************
'subroutine will generate random players
'*************************************************************************
Public Sub GenerateRandom()
    'variable to hold the number of potential opponents
    Dim intOpt As Integer
    Dim intX As Integer

    Dim intW As Integer
    Dim intS As Integer

    'array to store the available openent for each student
    Dim op(18) As String

    'read student records from the database
    getStudentArray strGstudent, "Green"
    getStudentArray strYstudent(), "Yellow"
add1:
    'clear games
    clearGameArray

    'snap columns in the game array are
    'green col =0
    'yellow col=1
    For intS = 0 To gintSnapMatch - 1
        Do
            Randomize (Timer)
                intX = Int(Rnd(Timer) * 18)
                If IsInGame(strGstudent(intX, 0), 0) = False Then
                    strGame(intS, 0) = strGstudent(intX, 0)
                    lblGSnapStudent(intS) = strGame(intS, 0)
                    Exit Do
                End If
        Loop
        'find potential opponents
        intOpt = potentialOponents(strGame(intS, 0), 1, op())

        'if no potential opponents were found restart the whole generation processes
        If intOpt = 0 Then GoTo add1
        'assign a random opponent
```

```
frmRandom - 11

        Randomize (Timer)
        strGame(intS, 1) = op(Int(Rnd(Timer) * intOpt))
        lblYSnapStudent(intS) = strGame(intS, 1)
    Next intS

    'cribbage columns in the game array are
    'green col =2
    'yellow col=3
    For intS = 0 To gintCribbageMatch - 1
        Do
            Randomize (Timer)
                intX = Int(Rnd(Timer) * 18)
                If IsInGame(strGstudent(intX, 0), 2) = False Then
                    strGame(intS, 2) = strGstudent(intX, 0)
                  lblGCribbageStudent(intS) = strGame(intS, 2)
                    Exit Do
                End If
        Loop
        'find potential opponents
        intOpt = potentialOponents(strGame(intS, 2), 3, op())
        'if no potential opponents were found restart the whole generation processes
        If intOpt = 0 Then GoTo add1
        'assign a random opponent
        Randomize (Timer)
        strGame(intS, 3) = op(Int(Rnd(Timer) * intOpt))
        lblYCribbageStudent(intS) = strGame(intS, 3)
    Next intS
    'spillikins columns in the game array are
    'green col =4
    'yellow col=5
    For intS = 0 To gintSpillikinsMatch - 1
        Do
            Randomize (Timer)
                intX = Int(Rnd(Timer) * 18)
                If IsInGame(strGstudent(intX, 0), 4) = False Then
                    strGame(intS, 4) = strGstudent(intX, 0)

                    lblGSpillikinsStudent(intS) = strGame(intS, 4)
                    Exit Do
                End If
        Loop
        'find potential opponents
        intOpt = potentialOponents(strGame(intS, 4), 5, op())
        'if no potential opponents were found restart the whole generation processes
        If intOpt = 0 Then GoTo add1
        'assign a random opponent
        Randomize (Timer)
        strGame(intS, 5) = op(Int(Rnd(Timer) * intOpt))
         lblYSpillikinsStudent(intS) = strGame(intS, 5)
    Next intS

    'scrabble columns in the game array are
    'green col =6
    'yellow col=7
    For intS = 0 To gintScrabbleMatch - 1
        Do
            Randomize (Timer)
                intX = Int(Rnd(Timer) * 18)
                If IsInGame(strGstudent(intX, 0), 6) = False Then
                    strGame(intS, 6) = strGstudent(intX, 0)
                    lblGScrabbleStudent(intS) = strGame(intS, 6)
                    Exit Do
                End If
        Loop
        'find potential opponents
        intOpt = potentialOponents(strGame(intS, 6), 7, op())
        'if no potential opponents were found restart the whole generation processes
        If intOpt = 0 Then GoTo add1
        'assign a random opponent
        Randomize (Timer)
        strGame(intS, 7) = op(Int(Rnd(Timer) * intOpt))
        lblYScrabbleStudent(intS) = strGame(intS, 7)
    Next intS

End Sub
```

frmRandom - 12

```
'*************************************************************************
'subroutine will display players selected manuallly
'*************************************************************************
Private Sub displayManual()
    Dim intX As Integer
    Dim intY As Integer
    For intX = 0 To 17
            lblGSnapStudent(intX).Caption = strGame(intX, 0)
    Next intX
    For intX = 0 To 17
            lblYSnapStudent(intX).Caption = strGame(intX, 1)
    Next intX
    For intX = 0 To 17
            lblGCribbageStudent(intX).Caption = strGame(intX, 2)
    Next intX
    For intX = 0 To 17
            lblYCribbageStudent(intX).Caption = strGame(intX, 3)
    Next intX
    For intX = 0 To 17
            lblGSpillikinsStudent(intX).Caption = strGame(intX, 4)
    Next intX
    For intX = 0 To 17
            lblYSpillikinsStudent(intX).Caption = strGame(intX, 5)
    Next intX
    For intX = 0 To 17
            lblGScrabbleStudent(intX).Caption = strGame(intX, 6)
    Next intX
    For intX = 0 To 17
            lblYScrabbleStudent(intX).Caption = strGame(intX, 7)
    Next intX
End Sub
```

5.11 Manual Games Schedule Generation Form

5.11.1 User interface (Task4/a)

This form is loaded from the Create Game window when the next command button is clicked with the manual option button selected. Or from the random generation form when the edit command button is clicked.

The combo boxes are updated automatically to allow the user to choose the student allowed playing in that position. In other words a list of all possible players in that position will be displayed in the combo box list.

This form is slightly different from the original design. I added the save command button to load the random generation form with the students selected and the results of each student can be read there.

I did not add text boxes to read the results as this can be done in the "random" form.

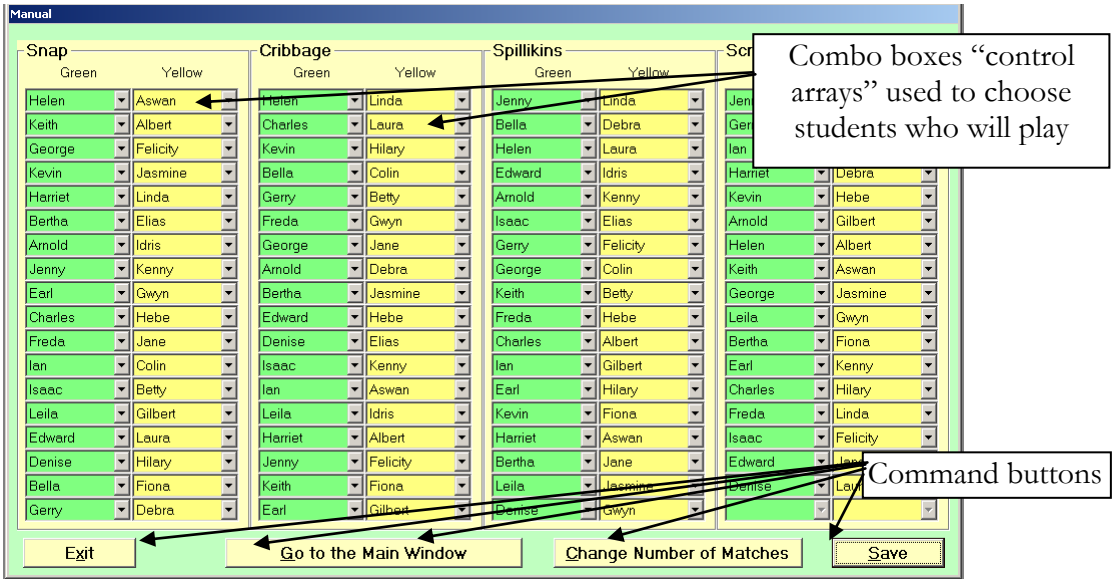


Figure 52

5.11.2 Testing (Task4/b)

To test each control, the correct data will be entered in all controls except the one that we are testing.

Test no.	Test	Test data	Expected Result	Actual Results	Comments
1	green students (combo boxes control array)	blank	When "Save" button is clicked a message will appear to ask the user to fill in all opponents in the game	When "Save" button was clicked a message appeared to ask the user to fill in all opponents in the game	Test passed
2	green students (combo boxes control array)	A student chosen	When "Save" is clicked , the random game form will be loaded showing the student selected	When "Save" was clicked, the random game form was loaded showing the student selected	Test passed
3	yellow students (combo boxes control array)	blank	When "Save" button is clicked a message will appear to ask the user to fill in all opponents in the game	When "Save" button was clicked a message appeared to ask the user to fill in all opponents in the game	Test passed
4	yellow students (combo boxes control array)	A student chosen	When "Save" is clicked , the random game form will be loaded showing the student selected	When "Save" was clicked, the random game form was loaded showing the student selected	Test passed
5	Exit Command Button	-	A message box will be displayed to ask the user to confirm if he/she want to exit the programme	A message box is displayed asking to confirm	Test passed
6	Go to main window button	-	The main window will be loaded	The main window was loaded	Test passed
7	Save command button	-	The random window will be loaded with the its labels displaying the students	The random window was loaded with the its labels displaying the students	Test passed

Table 24

5.11.3 Testing Evidence (Task4/b)

Evidence of this test is shown in the diagrams below; the test number refers to the test number field in table 23

Test no. 1

Number of matches: snap: 18, cribbage: 18, Spillikins: 18 and scrabble: 17

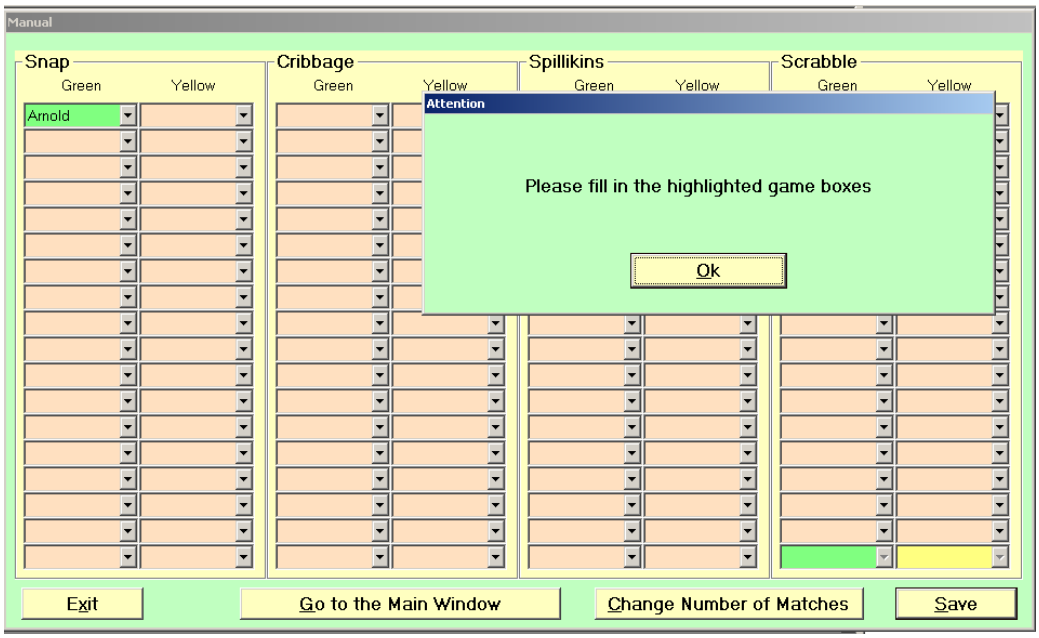


Figure 53

Test no. 7

Number of matches: snap: 1, cribbage: 1, Spillikins: 2 and scrabble: 1

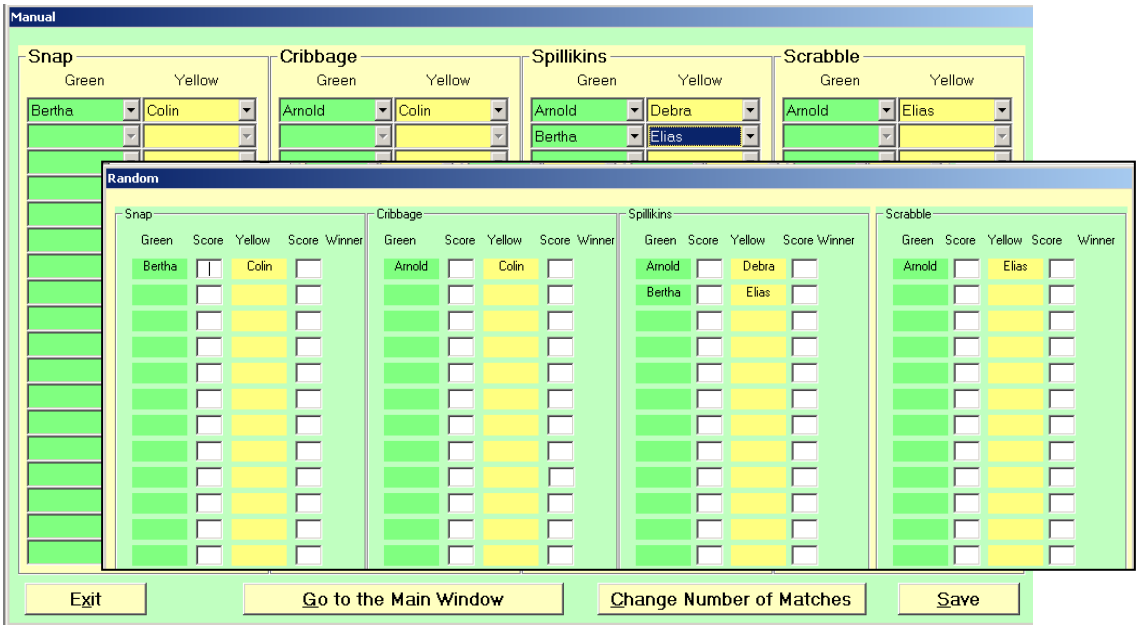


Figure 54(students selected in manual then saved in random forms)

Additional Test

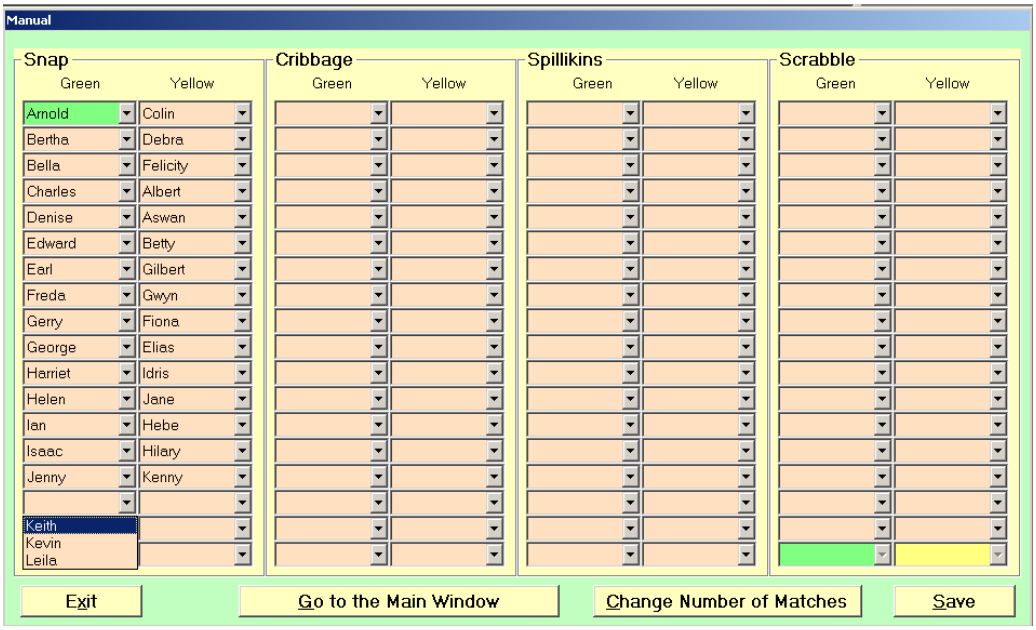


Figure 55(the combo box will only contain students allowed to play in that position)

5.11.4 Code (frmManual.frm) (Task4/a)

```
frmManual - 1

'*************************************************************************************
'************************** Tournament Organising System**************************
'*********************************frmManual Code*********************************
'*****************************Programer: S. Saqfelhait*************************
'*********************************Date:07/04/2007*********************************
'*************************************************************************************
'this form is loaded from frmIntro "select game generation method and number of
'matches form, it is also loaded from random form when Edit button is clicked
Option Explicit

'***************** Green Cribbage************************************************
'generate the list of players that are available to play
'*************************************************************************************
Private Sub cmbGCribbageStudent_DropDown(Index As Integer)

    Call Manual(cmbGCribbageStudent(Index), cmbYCribbageStudent(Index).Text, Index, 2)

End Sub

'***************** Green Cribbage************************************************
'update the game array and database once a player is selected
'*************************************************************************************
Private Sub cmbGCribbageStudent_Click(Index As Integer)

    If cmbGCribbageStudent(Index).ListIndex <> -1 Then
        gadoCommand.CommandText = "UPDATE Students  Set PlayedCribbage=true " _
        & "where studentName='" & cmbGCribbageStudent(Index).Text & "'"
        Set gadoRecordSet = gadoCommand.Execute
        strGame(Index, 2) = cmbGCribbageStudent(Index).Text
        cmbYCribbageStudent(Index).Enabled = True
    End If

End Sub
'***************** Green Scrabble************************************************
'generate the list of players that are available to play
'*************************************************************************************
Private Sub cmbGScrabbleStudent_DropDown(Index As Integer)

    Call Manual(cmbGScrabbleStudent(Index), cmbYScrabbleStudent(Index).Text, Index, 6)

End Sub
```

frmManual - 2

```
'***************** Green Scrabble*************************************************
'update the game array and database once a player is selected
'*********************************************************************************
Private Sub cmbGScrabbleStudent_Click(Index As Integer)

    If cmbGScrabbleStudent(Index).ListIndex <> -1 Then
        gadoCommand.CommandText = "UPDATE Students  Set PlayedScrabble=true " _
        & " where studentName='" & cmbGScrabbleStudent(Index).Text & "'"
        Set gadoRecordSet = gadoCommand.Execute
        strGame(Index, 6) = cmbGScrabbleStudent(Index).Text
        End If

End Sub

'***************** Green Snap*****************************************************
'generate the list of players that are available to play
'*********************************************************************************
Private Sub cmbGSnapStudent_DropDown(Index As Integer)

    Call Manual(cmbGSnapStudent(Index), cmbYSnapStudent(Index).Text, Index, 0)

End Sub

'***************** Green Snap*****************************************************
'update the game array and database once a player is selected
Private Sub cmbGSnapStudent_Click(Index As Integer)

  '  If cmbGSnapStudent(Index).ListIndex <> -1 Then
        gadoCommand.CommandText = "UPDATE Students  Set PlayedSnap=true " _
        & " where studentName='" & cmbGSnapStudent(Index).Text & "'"
        Set gadoRecordSet = gadoCommand.Execute
        strGame(Index, 0) = cmbGSnapStudent(Index).Text
   ' End If

End Sub

'***************** Green Spillikins**********************************************
'generate the list of players that are available to play
'*********************************************************************************
Private Sub cmbGSpillikinsStudent_DropDown(Index As Integer)

    Call Manual(cmbGSpillikinsStudent(Index), _
        cmbYSpillikinsStudent(Index).Text, Index, 4)

End Sub

'***************** Green Spillikins**********************************************
'update the game array and database once a player is selected
'*********************************************************************************
Private Sub cmbGSpillikinsStudent_Click(Index As Integer)
    If cmbGSpillikinsStudent(Index).ListIndex <> -1 Then
        gadoCommand.CommandText = "UPDATE Students  Set PlayedSpillikins=true " _
        & " where studentName='" & cmbGSpillikinsStudent(Index).Text & "'"
        Set gadoRecordSet = gadoCommand.Execute
        strGame(Index, 4) = cmbGSpillikinsStudent(Index).Text
    End If
End Sub

'***************** Yellow Cribbage***********************************************
'generate the list of players that are available to play
'*********************************************************************************
Private Sub cmbYCribbageStudent_DropDown(Index As Integer)
     Manual cmbYCribbageStudent(Index), cmbGCribbageStudent(Index).Text, Index, 3
End Sub

'***************** Yellow Cribbage***********************************************
'update the game array and database once a player is selected
'*********************************************************************************
Private Sub cmbYCribbageStudent_Click(Index As Integer)

    If cmbYCribbageStudent(Index).ListIndex <> -1 Then
        gadoCommand.CommandText = "UPDATE Students  Set PlayedCribbage=True " _
        & "where studentName='" & cmbYCribbageStudent(Index).Text & "'"
        Set gadoRecordSet = gadoCommand.Execute
```

```
frmManual - 3

        strGame(Index, 3) = cmbYCribbageStudent(Index).Text
    End If

End Sub

'***************** Yellow Scrabble**************************************************
'generate the list of players that are available to play
'*********************************************************************************
Private Sub cmbYScrabbleStudent_DropDown(Index As Integer)
 Manual cmbYScrabbleStudent(Index), cmbGScrabbleStudent(Index).Text, Index, 7

End Sub

'***************** Yellow Scrabble**************************************************
'update the game array and database once a player is selected
'*********************************************************************************
Private Sub cmbYScrabbleStudent_Click(Index As Integer)

    If cmbYScrabbleStudent(Index).ListIndex <> -1 Then
        gadoCommand.CommandText = "UPDATE Students  Set PlayedScrabble=True " _
        & "where studentName='" & cmbYScrabbleStudent(Index).Text & "'"
        Set gadoRecordSet = gadoCommand.Execute
        strGame(Index, 7) = cmbYScrabbleStudent(Index).Text
    End If

End Sub

'***************** Yellow Spillikins************************************************
'generate the list of players that are available to play
'*********************************************************************************
Private Sub cmbYSpillikinsStudent_DropDown(Index As Integer)

 Manual cmbYSpillikinsStudent(Index), cmbGSpillikinsStudent(Index).Text, Index, 5

End Sub

'***************** Yellow Spillikins************************************************
'update the game array and database once a player is selected
'*********************************************************************************
Private Sub cmbYSpillikinsStudent_Click(Index As Integer)

    If cmbYSpillikinsStudent(Index).ListIndex <> -1 Then
        gadoCommand.CommandText = "UPDATE Students  Set PlayedSpillikins=True " _
        & "where studentName='" & cmbYSpillikinsStudent(Index).Text & "'"
        Set gadoRecordSet = gadoCommand.Execute
        strGame(Index, 5) = cmbYSpillikinsStudent(Index).Text
    End If

End Sub

'***************** Yellow Snap******************************************************
'generate the list of players that are available to play
'*********************************************************************************
Private Sub cmbYSnapStudent_DropDown(Index As Integer)

   Manual cmbYSnapStudent(Index), cmbGSnapStudent(Index).Text, Index, 1

End Sub

'***************** Yellow Snap******************************************************
'update the game array and database once a player is selected
'*********************************************************************************
Private Sub cmbYSnapStudent_Click(Index As Integer)

    If cmbYSnapStudent(Index).ListIndex <> -1 Then
        gadoCommand.CommandText = "UPDATE Students  Set PlayedSnap=True " _
        & "where studentName='" & cmbYSnapStudent(Index).Text & "'"
        Set gadoRecordSet = gadoCommand.Execute
        strGame(Index, 1) = cmbYSnapStudent(Index).Text
    End If

End Sub
'*********************************************************************************
'subroutine will be executed when the Go to main window command button is clicked
'*********************************************************************************
```

frmManual - 4

```
Private Sub cmdMain_Click()
    frmMain.Show
    Unload Me
End Sub

'*************************************************************************
'subroutine will be executed when the Change Number of matches command button is clicked
'*************************************************************************
Private Sub cmdChange_Click()
    frmIntro.Show
    Unload Me
End Sub

'*************************************************************************
'subroutine will be executed when the Exit command button is clicked
'*************************************************************************
Private Sub cmdExit_Click()

  'define a variable to capture the msgbox response
    Dim varResponce As Variant
    varResponce = myMsgBox("Are You Sure you Want to Exit", "YesNo", "Attention")
    'exit program if response if Yes
    If varResponce = vbYes Then
        DB_Disconnect
        End
    End If
End Sub

'*************************************************************************
'subroutine will be executed when the Save command button is clicked
'*************************************************************************
Private Sub cmdSave_Click()
    'define a variable "flag" which is set to true if any
    'enabled combo box is empty and initialise it to false
    Dim emptyResults As Boolean
    emptyResults = False

    Dim intX As Integer
    'check if any enabled combo box is empty
    'Snap
    For intX = 0 To gintSnapMatch - 1
        If cmbGSnapStudent(intX).Text = "" Then
            cmbGSnapStudent(intX).BackColor = &HC0E0FF
            emptyResults = True
        End If

        If cmbYSnapStudent(intX).Text = "" Then
            cmbYSnapStudent(intX).BackColor = &HC0E0FF
            emptyResults = True
        End If
    Next intX
    'Cribbage
    For intX = 0 To gintCribbageMatch - 1
        If cmbGCribbageStudent(intX).Text = "" Then
            cmbGCribbageStudent(intX).BackColor = &HC0E0FF
            emptyResults = True
        End If

        If cmbYCribbageStudent(intX).Text = "" Then
            cmbYCribbageStudent(intX).BackColor = &HC0E0FF
            emptyResults = True
        End If
    Next intX
    'Spillikins
    For intX = 0 To gintSpillikinsMatch - 1
        If cmbGSpillikinsStudent(intX).Text = "" Then
            cmbGSpillikinsStudent(intX).BackColor = &HC0E0FF
            emptyResults = True
        End If

        If cmbYSpillikinsStudent(intX).Text = "" Then
            cmbYSpillikinsStudent(intX).BackColor = &HC0E0FF
            emptyResults = True
        End If
    Next intX
```

frmManual - 5

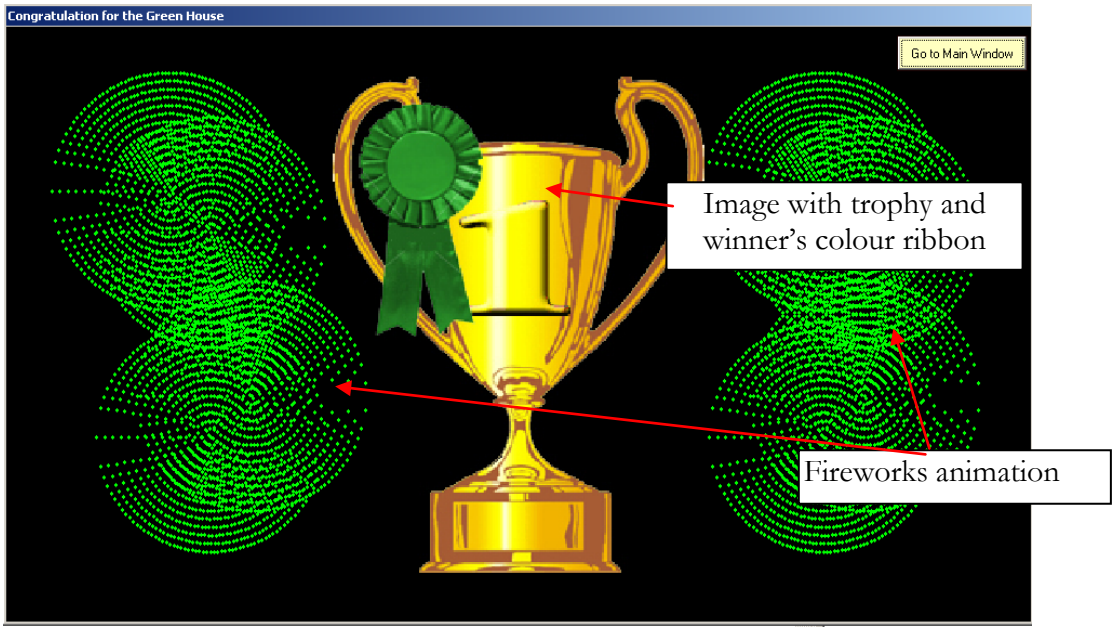
```
    'Scrabble
    For intX = 0 To gintScrabbleMatch - 1
        If cmbGScrabbleStudent(intX).Text = "" Then
            cmbGScrabbleStudent(intX).BackColor = &HC0E0FF
            emptyResults = True
        End If

        If cmbYScrabbleStudent(intX).Text = "" Then
            cmbYScrabbleStudent(intX).BackColor = &HC0E0FF
            emptyResults = True
        End If
    Next intX

    If emptyResults = True Then
    myMsgBox "Please fill in the highlighted game boxes", "Ok", "Attention"
    Exit Sub
    End If

    'save and load the random form with the selected games
    gboolRandom = False
    frmRandom.Show
    Unload Me
End Sub
'*************************************************************************
'subroutine will be executed when the form is loaded
'*************************************************************************
Private Sub Form_Load()
  Dim intY As Integer
    If gboolEdit = True Then

        For intY = 0 To gintSnapMatch - 1
            cmbGSnapStudent(intY).AddItem strGame(intY, 0)
            cmbGSnapStudent(intY).ListIndex = 0
            cmbYSnapStudent(intY).AddItem strGame(intY, 1)
            cmbYSnapStudent(intY).ListIndex = 0
        Next intY

        For intY = 0 To gintCribbageMatch - 1
            cmbGCribbageStudent(intY).AddItem strGame(intY, 2)
            cmbGCribbageStudent(intY).ListIndex = 0
            cmbYCribbageStudent(intY).AddItem strGame(intY, 3)
            cmbYCribbageStudent(intY).ListIndex = 0
        Next intY

        For intY = 0 To gintSpillikinsMatch - 1
            cmbGSpillikinsStudent(intY).AddItem strGame(intY, 4)
            cmbGSpillikinsStudent(intY).ListIndex = 0
            cmbYSpillikinsStudent(intY).AddItem strGame(intY, 5)
            cmbYSpillikinsStudent(intY).ListIndex = 0
        Next intY

        For intY = 0 To gintScrabbleMatch - 1
            cmbGScrabbleStudent(intY).AddItem strGame(intY, 6)
            cmbGScrabbleStudent(intY).ListIndex = 0
            cmbYScrabbleStudent(intY).AddItem strGame(intY, 7)
            cmbYScrabbleStudent(intY).ListIndex = 0
        Next intY

    Else
        'clear games
        clearGameArray
        'clear the game schedule in the database
        clearDatabase
        'read students list from the database
        getStudentArray strGstudent, "Green"
        getStudentArray strYstudent(), "Yellow"
    End If

    'enable the combo boxes based on the number of matches
    For intY = gintSnapMatch To 17
        cmbGSnapStudent(intY).Enabled = False
        cmbYSnapStudent(intY).Enabled = False

    Next intY
    For intY = gintCribbageMatch To 17
```

```
frmManual - 6

        cmbGCribbageStudent(intY).Enabled = False
        cmbYCribbageStudent(intY).Enabled = False

    Next intY
    For intY = gintSpillikinsMatch To 17
        cmbGSpillikinsStudent(intY).Enabled = False
        cmbYSpillikinsStudent(intY).Enabled = False

    Next intY
    For intY = gintScrabbleMatch To 17
        cmbGScrabbleStudent(intY).Enabled = False
        cmbYScrabbleStudent(intY).Enabled = False

    Next intY

End Sub
```

5.12 Winner Form

5.12.1 User interface (Task4/a)

This form is loaded from the Random Game Schedule Generation Form when the Winner command button is clicked. The purpose of this form is to greet the winning house. A trophy image with a ribbon of the winning house colour will be displayed. Animation of fireworks using the winning house colours will also be shown.



I used timers to allow the animation. The code is shown below

```
'*************************************************************************
'subroutine will be executed when the timer is enabled and every  "interval"
'*************************************************************************
Private Sub tmrExplosion_Timer()
'if no circles draw else erase
    If gErase = "Winner" Then
        drawCircle 50, 50, intR
        drawCircle 210, 50, intR

        intR = intR + 2
    ElseIf gErase = "Var" Then
        varCircle 50, 50, intR
        varCircle 210, 50, intR
    Else
        eraseCircle 50, 50, intR
        eraseCircle 210, 50, intR
    End If
    'when the max radius is reached change status from erase to draw
    'or from draw to erase
    If intR >= 30 Then
        intR = 0
        If gErase = "Winner" Then
            gErase = "Var"
        ElseIf gErase = "Var" Then
            gErase = "Erase"
        Else
            gErase = "Winner"
        End If
    End If
End Sub

'*************************************************************************
'subroutine, draw a dotted circle with the current forecolor
'input:X- cordinate of the center of the circle:intX
'input:Y- cordinate of the center of the circle:intY
'input:the raduis of the circle:intRad
'*************************************************************************
Private Sub drawCircle(intX As Integer, intY As Integer, intRad As Integer)
    Dim intZ As Double

    For intZ = -intRad To intRad Step 0.9
        CurrentX = intZ + intX
        CurrentY = Abs(Sqr(Abs((intRad * intRad) - (intZ * intZ)))) + intY
        frmWinner.PSet (CurrentX, CurrentY)
    Next intZ
    For intZ = -intRad To intRad Step 0.9
        CurrentX = intZ + intX
        CurrentY = -Abs(Sqr(Abs((intRad * intRad) - (intZ * intZ)))) + intY
        frmWinner.PSet (CurrentX, CurrentY)
    Next intZ
End Sub
```

5.12.2 Testing (Task4/b)

To test each control, the correct data will be entered in all controls except the one that we are testing.

Test no.	Test	Test data	Expected Result	Actual Results	Comments
1	Form loaded when green wins	-	The colour of the trophy ribbon and the fire work will be green	The colour of the trophy ribbon and the fire work is green	Test passed
2	Form loaded when yellow wins	-	The colour of the trophy ribbon and the fire work will be yellow	The colour of the trophy ribbon and the fire work is yellow	Test passed
3	Go to main window button	-	The main window will be loaded	The main window was loaded	Test passed

Table 25

5.12.3 Code (frmWinner.frm) (Task4/a)

```
frmWinner - 1

'************************** Tournament Organising System**********************
'*********************************frmWinner Code*****************************
'****************************Programer: Somoud Saqfelhait**********************
'*********************************Date:07/06/2007****************************
'****************************************************************************
'this form will be loaded from the random form when winner command button is clicked

Option Explicit
Dim gErase As String
'circle radius
Dim intR As Integer

Private Sub cmdMain_Click()
    Unload Me
    frmMain.Show
End Sub

'****************************************************************************
'subroutine will be executed when the form is activated
'****************************************************************************
Private Sub Form_Activate()
    gErase = "Winner"
    picTrophy.Left = (Me.ScaleWidth - picTrophy.Width) / 2
    picTrophy.Top = (Me.ScaleHeight - picTrophy.Height) / 2
    picTrophy.Width = 1
    picTrophy.Height = 1
End Sub

'****************************************************************************
'subroutine will be executed when the form is de-activated
'****************************************************************************
Private Sub Form_Deactivate()
    tmrExplosion.Enabled = False

End Sub

'****************************************************************************
'subroutine will be executed when the form is loaded
'****************************************************************************
Private Sub Form_Load()
    If gstrWinner = "Green" Then
        ForeColor = vbGreen
        'load the picture of the trophy with green rippon
        picTrophy.Picture = LoadPicture(App.Path & "\greentrophy.jpg")
        cmdMain.BackColor = &HC0FFFF
    ElseIf gstrWinner = "Yellow" Then
        ForeColor = vbYellow
        cmdMain.BackColor = &HC0FFC0
        'load the picture of the trophy with yellow rippon
        picTrophy.Picture = LoadPicture(App.Path & "\yellowtrophy.jpg")
    Else
        ForeColor = &HC0FFC0
    End If
    Caption = Caption + " for the " + gstrWinner + " House"

End Sub

'****************************************************************************
'subroutine will be executed when the form is clicked
'****************************************************************************
Sub Form_Click()
  Unload Me
  frmMain.Show
End Sub

'****************************************************************************
'subroutine will be executed when the timer is enabled and every  "interval"
'****************************************************************************
Private Sub tmrExplosion_Timer()
'if no circles draw else erase
 If gErase = "Winner" Then
        drawCircle 50, 50, intR
        drawCircle 210, 50, intR
```

```
frmWinner - 2

        drawCircle 40, 40, intR + 1
        drawCircle 200, 40, intR + 1
        drawCircle 200, 100, intR
        drawCircle 50, 100, intR
        drawCircle 210, 90, intR
        drawCircle 60, 90, intR
        intR = intR + 2
    ElseIf gErase = "Var" Then
        varCircle 50, 50, intR
        varCircle 210, 50, intR
        varCircle 40, 40, intR + 1
        varCircle 200, 40, intR + 1
        varCircle 200, 100, intR
        varCircle 50, 100, intR
        varCircle 210, 90, intR
        varCircle 60, 90, intR
        intR = intR + 2
    Else
        eraseCircle 50, 50, intR
        eraseCircle 210, 50, intR
        eraseCircle 40, 40, intR + 1
        eraseCircle 200, 40, intR + 1
        eraseCircle 200, 100, intR
        eraseCircle 50, 100, intR
        eraseCircle 210, 90, intR
        eraseCircle 60, 90, intR
        intR = intR + 2
    End If
    'when the max radius is reached change status from erase to draw
    'or from draw to erase
    If intR >= 30 Then
        intR = 0
        If gErase = "Winner" Then
            gErase = "Var"
        ElseIf gErase = "Var" Then
            gErase = "Erase"
        Else
            gErase = "Winner"
        End If
    End If
End Sub

'*************************************************************************
'subroutine will be executed when the timer is enabled and every  "interval"
'*************************************************************************
Private Sub tmrTrophy_Timer()
'while the width is less than 80 increase the pic dimensions
    If picTrophy.Width > 80 Then
        tmrExplosion.Enabled = True
        tmrTrophy.Enabled = False
    End If
    picTrophy.Width = picTrophy.Width + 6
    picTrophy.Height = picTrophy.Height + 8
    'keep the pic centered
    picTrophy.Left = (Me.ScaleWidth - picTrophy.Width) / 2
    picTrophy.Top = (Me.ScaleHeight - picTrophy.Height) / 2
End Sub

'*************************************************************************
'subroutine, draw a dotted circle with the current forecolor
'input:X- cordinate of the center of the circle:intX
'input:Y- cordinate of the center of the circle:intY
'input:the raduis of the circle:intRad
'*************************************************************************
Private Sub drawCircle(intX As Integer, intY As Integer, intRad As Integer)
    Dim intZ As Double

    For intZ = -intRad To intRad Step 0.9
        CurrentX = intZ + intX
        CurrentY = Abs(Sqr(Abs((intRad * intRad) - (intZ * intZ)))) + intY
        frmWinner.PSet (CurrentX, CurrentY)
    Next intZ
    For intZ = -intRad To intRad Step 0.9
```

```
frmWinner - 3

        CurrentX = intZ + intX
        CurrentY = -Abs(Sqr(Abs((intRad * intRad) - (intZ * intZ)))) + intY
        frmWinner.PSet (CurrentX, CurrentY)
    Next intZ
End Sub

'*************************************************************************
'subroutine, draw a dotted circle with random colour
'input:X- cordinate of the center of the circle:intX
'input:Y- cordinate of the center of the circle:intY
'input:the raduis of the circle:intRad
'*************************************************************************
Private Sub varCircle(intX As Integer, intY As Integer, intRad As Integer)
    Dim intZ As Double
    Dim temp As Variant
    temp = ForeColor
    ForeColor = QBColor((Rnd * 10) + 5)
    For intZ = -intRad To intRad Step 0.9
        CurrentX = intZ + intX
        CurrentY = Abs(Sqr(Abs((intRad * intRad) - (intZ * intZ)))) + intY
        frmWinner.PSet (CurrentX, CurrentY)
    Next intZ
    For intZ = -intRad To intRad Step 0.9
        CurrentX = intZ + intX
        CurrentY = -Abs(Sqr(Abs((intRad * intRad) - (intZ * intZ)))) + intY
        frmWinner.PSet (CurrentX, CurrentY)
    Next intZ
    ForeColor = temp
End Sub

'*************************************************************************
'subroutine, draw a dotted circle with the current backcolor i.e erase any
'circle with the same coordinates
'input:X- cordinate of the center of the circle:intX
'input:Y- cordinate of the center of the circle:intY
'input:the raduis of the circle:intRad
'*************************************************************************
Private Sub eraseCircle(intX As Integer, intY As Integer, intRad As Integer)
    Dim intZ As Double
    Dim temp As Variant
    temp = ForeColor
    ForeColor = BackColor
    For intZ = -intRad To intRad Step 0.9
        CurrentX = intZ + intX
        CurrentY = Abs(Sqr(Abs((intRad * intRad) - (intZ * intZ)))) + intY
        frmWinner.PSet (CurrentX, CurrentY)
    Next intZ
    For intZ = -intRad To intRad Step 0.9
        CurrentX = intZ + intX
        CurrentY = -Abs(Sqr(Abs((intRad * intRad) - (intZ * intZ)))) + intY
        frmWinner.PSet (CurrentX, CurrentY)
    Next intZ
    ForeColor = temp
End Sub
```

6 Evaluation (Task5)

6.1 Effectiveness of Programming Environment (Task5/a)

I have used **V**isual **B**asic (VB) to develop the **T**ournament **O**rganising **S**ystem (TOS). It is an event driven programming language. It enables **R**apid **A**pplication **D**evelopment (RAD) of **G**raphical **U**ser **I**nterface (GUI) which allows fast implementation of programme's user interface. VB provides access to database using Data Access Object (DAO), **R**emote **D**ata **O**bjects (RDO) and (**A**ctiveX **D**ata **O**bjects) ADO.
Visual Basic is designed to be easy to learn and use. It allows the creation of simple GUI. It also has the flexibility to develop fairly complex applications. [3]

With the use of native code compilation and faster computers, performance problems experienced in earlier versions of VB are less of an issue. However, for applications that contain huge calculations, this is still an issue.
I faced performance problem when I implemented the first version of random game generation algorithm for the maximum number of matches for each game (18 each). Because this version of the algorithm relied on selecting a player randomly and test if he/she satisfies the game rules. In other words the programme would go into a loop until it generates a number corresponding to a student who satisfies the game rules.
With VB being slow compared to some other programming languages such as C++, this process took a long processing time.
However, given that C++ MFC is complicated and needs a lot of coding to perform operations that can be implemented in VB with a line of code or no code at all. I preferred to stick to using VB and tried to minimise the processing time required to generate games schedule.
The solution was to force the programme to randomly select a player who satisfies the game rules. I wrote a function that generates a list of students who are allowed to play a specific student. The function checks students who already played, students who already played each other and the class of the student.
I used this function in both random and manual game generation. The only difference is that, in random generation, the programme will choose the student from the potential

players list. While in manual generation, the potential players list will be displayed in the combo box to the user to choose from.

The above method allowed a great reduction in the time needed to generate the games schedule. The only problem that could occur is; based on the students chosen at earlier stage, there might not remain any potential players.

In random generation case, the process would have to be restarted from the very beginning until each match is assigned the opponents who satisfy the game rules. The processing time is much less than that of the initial algorithm and not noticeable for users using hardware and software that complies with the optimal system requirements.

In manual generation case, there will be no student names displayed in the combo box, so the user might change some earlier choices till each match is assigned the opponents who satisfy the game rules. A message box will inform the user what to do

I have used ADO to connect the system to the Microsoft Access Database "students.mdb"; in which I stored each student's name, class and house. The database was used to store and retrieve information about each student's participation in the tournaments. Connecting to database, retrieving, storing and even querying is a very easy straight forward operation which requires only a minimal knowledge of **S**equential **Q**uery **L**anguage (SQL). For example in the students statistics form I used the SQL query "Order by" to sort the names or the numbers. If I was to implement this using C++ I would have to write a lot of code to perform the same task

```
"SELECT * FROM statistics ORDER BY SudentName 'ASC'"
```

From all of the above we can see that the programming environment that I have selected (Visual Basic) is appropriate for the purpose of developing the tournament organising system.

Using a traditional programming language such as C, C++ or Pascal is not appropriate for implementing this system, because most applications developed using these programming languages need special training for the system users. In addition to the complexity and time added to the development processes. The following comparison between traditional and visual environment in programming clarifies this issue. [3]

6.1.1 User benefits

The major benefit of a visual environment to the user is that it is much more closely aligned to what the user wants to do. Therefore it makes it easier to use, often much easier than a non-visual programme.
There are many features that are very difficult if not impossible, to programme in a non-visual environment.

Visual output can be programmed in traditional way. Visual input when it is expected can also be programmed in the traditional way. The benefit of a visual environment, however, comes in handling unexpected and uncontrolled user input.

Visually interfaced programs are much easier to use since they tend to have user friendly controls, any user with the minimised knowledge of computers and software would know how to use a pushbutton, a check box or a drop-in list [1]

6.1.2 Developer (programmer) Benefits

From a developers point of view, the benefits of a visual environment comes from having an environments where complexities are automatically controlled

- Handling graphics
- Managing unexpected interaction with the user
- Controlling the graphical run-time environment
- Linking input screens and output screens together

These benefits are gained through the use of objects, classes and inheritance. The designer of the software environment provides a template to how a visual programme works and makes that available to the developer. This template handles all of the complexity in handling a visual environment. The benefit of this to the developer is that he or she can concentrate on the just the problem in hand or the programme he or she is trying to write. The developer only needs to change the way the standard programme works if that is necessary to solve the problem. In general this will minimise the time needed to develop the visual programs.[1]

6.2 Effectiveness of Programme and Testing Procedures (Task5/b)

6.2.1 Design Techniques

The design technique I used in designing and implementing TOS is the top-down "waterfall" design model, which emphasises planning and a complete understanding of the system. It is inherent that no coding can begin until a sufficient level of detail has been reached in the design of at least some part of the system. This, however, delays testing of the ultimate functional units of a system until significant design is complete. [3] **R**apid **A**pplication **D**evelopment (RAD) design techniques, which are based on developing a simple version of the programme and getting user feedback on the system through out the development process, will be a better alternative to top-down design as it saves time and produce a software that satisfies user requirements. [4]

6.2.2 Programme Analysis

Implementing the programme after the design is completed; it was easy to follow the design as well as doing some changes.

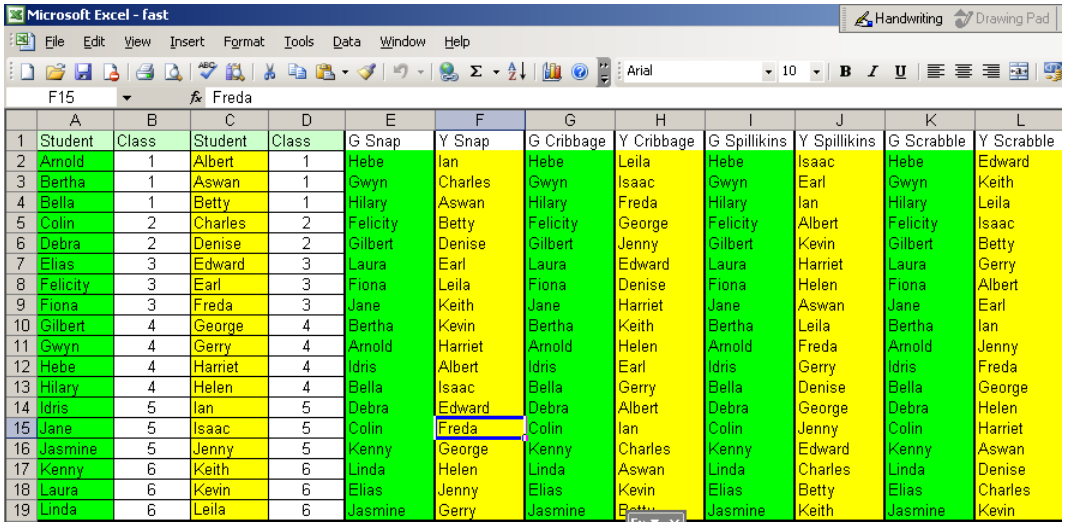
I used a two-dimensional strings array to store the matches of the game during the running time. An alternative is to use a one-dimensional array of data structures as shown in the code below.

```
Public Type StudentRecord
    intStudentID As Integer 'student ID, unique number
    strStudentName As String 'student Name
    strHouse As String 'the student house
    strClass As String 'the student class
    boolPlaySnap As Boolean 'true if student is already selected to play snap
    boolPlayCribbage As Boolean 'true if student is already selected to play cribbage
    boolPlaySpillikins As Boolean 'true if student is already selected to play spillikins
    boolPlayScrabble As Boolean 'true if student is already selected to play scrabble
    intSnapOponent As Integer  'store the student ID of openent in snap
    intCribbageOponent As Integer  'store the student ID of openent in cribbage
    intSpillikinsOponent As Integer  'store the student ID of openent in spillikins
    intScrabbleOponent As Integer  'store the student ID of openent in scrabble
End Type
'create a data type to hold game info
Public Type game
    strGstudent As StudentRecord  'first player Green
    strYstudent As StudentRecord  'the oponent Yellow
    strResult As String 'game result true if green wins
End Type
'define an array to hold the three snap games details
Public ggSnap(3) As game
'define an array to hold the three cribbage games details
Public ggCribbage(3) As game
'define an array to hold the three spillikin games details
Public ggSpillikins(3) As game
'define an array to hold the three scrabble games details
Public ggScrabble(3) As game
```

Using the structure type above would have made the process of generating the games schedule faster. That is because less time will be taken in search for a student who satisfies the rules. Only one loop will be required, while in the method I implemented I needed to use nested loops "in the state of functions" for the same search process. Refer to random form code and module code for the complete code.

6.2.3 Testing procedures

Using a prototype designed in Microsoft Excel I was able to test the algorithm for random games generation, view the speed of generation and make sure that the algorithm used satisfied the games rules stated in the user requirements. The last was achieved by displaying the class's numbers of opponents instead of the names (using Vlookup in excel) as shown below



	A	B	C	D	E	F	G	H	I	J	K	L
1	Student	Class	Student	Class	G Snap	Y Snap	G Cribbage	Y Cribbage	G Spillikins	Y Spillikins	G Scrabble	Y Scrabble
2	Arnold	1	Albert	1	Hebe	Ian	Hebe	Leila	Hebe	Isaac	Hebe	Edward
3	Bertha	1	Aswan	1	Gwyn	Charles	Gwyn	Isaac	Gwyn	Earl	Gwyn	Keith
4	Bella	1	Betty	1	Hilary	Aswan	Hilary	Freda	Hilary	Ian	Hilary	Leila
5	Colin	2	Charles	2	Felicity	Betty	Felicity	George	Felicity	Albert	Felicity	Isaac
6	Debra	2	Denise	2	Gilbert	Denise	Gilbert	Jenny	Gilbert	Kevin	Gilbert	Betty
7	Elias	3	Edward	3	Laura	Earl	Laura	Edward	Laura	Harriet	Laura	Gerry
8	Felicity	3	Earl	3	Fiona	Leila	Fiona	Denise	Fiona	Helen	Fiona	Albert
9	Fiona	3	Freda	3	Jane	Keith	Jane	Harriet	Jane	Aswan	Jane	Earl
10	Gilbert	4	George	4	Bertha	Kevin	Bertha	Keith	Bertha	Leila	Bertha	Ian
11	Gwyn	4	Gerry	4	Arnold	Harriet	Arnold	Helen	Arnold	Freda	Arnold	Jenny
12	Hebe	4	Harriet	4	Idris	Albert	Idris	Earl	Idris	Gerry	Idris	Freda
13	Hilary	4	Helen	4	Bella	Isaac	Bella	Gerry	Bella	Denise	Bella	George
14	Idris	5	Ian	5	Debra	Edward	Debra	Albert	Debra	George	Debra	Helen
15	Jane	5	Isaac	5	Colin	Freda	Colin	Ian	Colin	Jenny	Colin	Harriet
16	Jasmine	5	Jenny	5	Kenny	George	Kenny	Charles	Kenny	Edward	Kenny	Aswan
17	Kenny	6	Keith	6	Linda	Helen	Linda	Aswan	Linda	Charles	Linda	Denise
18	Laura	6	Kevin	6	Elias	Jenny	Elias	Kevin	Elias	Betty	Elias	Charles
19	Linda	6	Leila	6	Jasmine	Gerry	Jasmine	Betty	Jasmine	Keith	Jasmine	Kevin

Figure 56(Excel Prototype, opponents)

G Snap	Y Snap	Y Cribbage	Y Spillikins	Y Scrabble
4	1	6	3	2
1	3	6	4	6
4	1	1	5	1
3	5	4	6	4
2	6	5	5	4
5	4	2	6	4
6	2	5	5	4
2	4	1	3	3
4	5	3	1	1
3	6	4	2	5
6	3	5	2	5
4	2	3	1	6
1	5	6	6	3
6	4	2	1	5
5	6	4	3	1
5	1	3	4	3
1	3	4	4	6
3	4	1	4	2

Figure 57(Excel prototype, comparison of opponents classes)

The prototype in Excel also helped defining the problems that might be encountered in the generation process; i.e. when no potential players are left. See section 6.1 for more details

The initial implementation relied on the use of the database to flag students when they play. Information would be stored about the opponent in each game and weather the game has been played or not. This approach meant that the program continuously needs to communicate with the database and consequently the processing time required to generate the game would increase, see section 6.1 for more details

The use of the game 2-dimensional array has solved the above problem, but of course it needed extra code to be written.

7 Bibliography

1. BTEC National, IT Practitioners, Software Development, Jenny Lawson et al.
2. Computer Science, CS French, fifth edition.
3. http://www.wikipedia.org
4. http://www.21stsoft.com/services-methodologies-rad.asp