Technical Documentation

Child Education System

Github: https://github.com/IkyaVenus/FunBrainz-Assignment-3/tree/master/FunBrainz/FunBrainz

This software is developed on Microsoft Visual Studio 2010 and written using visual c++ and MS Access Database.

Common libraries

#pragma once **:** In the C and C++ programming languages, It is a non-standard but widely supported preprocessor directive designed to cause the current source file to be included only once in a single compilation.

#include <stdio.h> : STDIO.H is a file which contain declaration of many functions and Macros which required to get input from input devices and show output on output screen of C++ Program.

#include <utility> : It is a header file which contains utilities in unrelated domains.

- Pairs: Theses are the objects which can hold two different types of values.
- Generic Relational Approach: It is used for the relational operators !=, >, = under a specific namespace: rel_ops.
- Generic swap function: This a standard definition used by default by the components of the standard library for all types that do not provide their own overload: swap.

#include<cmath> : Header <cmath> declares a set of functions to compute common mathematical operations and transformations

#include<ctime> : "#include <ctime>" tells the linker "look in the ctime library for functions." It does and finds the function "time" and from there it compiles and links.

#include<string> : The string class is part of the C++ standard library. A string represents a sequence of characters.

To use the string class, #include the header file:

#include <string> :

Constructors:

- string ()
  - creates an empty string ("")
- string (other_string)
  - creates a string identical to other_string
- string (other_string, position, count)
  - creates a string that contains count characters from other_string, starting at position. If count is missing (only the first two arguments are given), all the characters from other_string, starting at position and going to the end of other_string, are included in the new string.
- string (count, character)
  - create a string containing character repeated count times

#include<iostream> : It is the predefined library function used for input and output also called as header files. iostream is the header file which contains all the functions of program like cout, cin etc. and #include tells the preprocessor to include these header file in the program.

#include <fstream> : Stream class to both read and write from/to files.

#include <vector> : Vectors are same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container. Vector elements are placed in contiguous storage so that they can be accessed and traversed using iterators. In vectors, data is inserted at the end. Inserting at the end takes differential time, as sometimes there may be a need of extending the array. Removing the last element takes only constant time because no resizing happens. Inserting and erasing at the beginning or in the middle is linear in time.

Certain functions associated with the vector are:
Iterators

1. begin() – Returns an iterator pointing to the first element in the vector
2. end() – Returns an iterator pointing to the theoretical element that follows the last element in the vector
3. rbegin() – Returns a reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element

4. rend() – Returns a reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end)
5. cbegin() – Returns a constant iterator pointing to the first element in the vector.
6. cend() – Returns a constant iterator pointing to the theoretical element that follows the last element in the vector.
7. crbegin() – Returns a constant reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element
8. crend() – Returns a constant reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end)

#include <Windows.h> : <windows.h> header file is used to access the Win32 API functions and it makes it easier for the user to use the in-built functionality.

- The header file in particular includes the library and functions used in the libraries like stdio.h or stdlib.h.

- It includes the functionality of macros and other components are used that provide modification, extension and replace of the things in the libraries.

#include <vcclr.h> : Forward declaration of C# tyes in managed C++ headers.

#include <iterator> : Iterators are used to point at the memory addresses of STL containers. They are primarily used in sequence of numbers, characters etc. They reduce the complexity and execution time of program.

Operations of iterators :-

1. begin() :- This function is used to return the **beginning position** of the container.

2. end() :- This function is used to return the ***after* end position** of the container.

#include<algorithm> : For all those who aspire to excel in competitive programming, only having a knowledge about containers of STL is of less use till one is not aware what all STL has to offer.
STL has an ocean of algorithms, for all < algorithm > library functions.
Some of the most used algorithms on vectors and most useful one's in Competitive Programming are mentioned as follows :

Non-Manipulating Algorithms

1. **sort(first_iterator, last_iterator)** – To sort the given vector.

2. **reverse(first_iterator, last_iterator)** – To reverse a vector.
3. ***max_element (first_iterator, last_iterator)** – To find the maximum element of a vector.
4. ***min_element (first_iterator, last_iterator)** – To find the minimum element of a vector.
5. **accumulate(first_iterator, last_iterator, initial value of sum)** – Does the summation of vector elements

Common windows properties:

Using namespace System:System Namespace : The System namespace contains fundamental classes and base classes that define commonly-used value and reference data types, events and event handlers, interfaces, attributes, and processing exceptions.

Using namespace System::ComponentModel:System.ComponentModel Namespace. The System.ComponentModel namespace provides classes that are used to implement the run-time and design-time behavior of components and controls.

Using namespace System::Collections : System.Collections Namespace. The System.Collections namespace contains interfaces and classes that define various collections of objects, such as lists, queues, bit arrays, hash tables and dictionaries.

Using namespace System::Windows::Forms : The System.Windows.Forms namespace contains classes for creating Windows-based applications that take full advantage of the rich user interface features available in the Microsoft Windows operating system.

Using namespace System::Data : The System.Data namespace provides access to classes that represent the ADO.NET architecture. ADO.NET lets you build components that efficiently manage data from multiple data sources.

Using namespace System::Drawing : The System.Drawing namespace provides access to GDI+ basic graphics functionality. More advanced functionality is provided in the System.Drawing.Drawing2D, System.Drawing.Imaging, and System.Drawing.Text namespaces.

InitializeComponent() : InitializeComponent is a method automatically written for you by the Form Designer when you create/change your forms.

Every Forms file (e.g. Form1.cs) has a designer file (e.g. Form1.designer.cs) that contains the InitializeComponent method, the override of the generic Form.Dispose, and the declaration of all of your User Interface objects like buttons, textboxes, labels and the Form itself.

The InitializeComponent method contains the code that creates and initializes the user interface objects dragged on the form surface with the values provided by you (the programmer) using the Property Grid of the Form Designer. Due to this fact do not ever try to interact with the form or the controls before the call to InitializeComponent.

#pragma region Windows Form Designer generated code : The components member is generated as part of the **"**Windows Forms Designer generated code**"** region, which is part of every Form created and managed by means of the Visual Studio .NET Windows Forms designer:

using namespace System::Data::OleDb : The System.Data.OleDb Namespace. The System.Data.OleDb namespacecontains the provider-specific ADO.NET classes used to connect to an OLE DB data source, execute a command, and transfer data to and from aSystem.Data.DataSet .

using namespace System::Diagnostics : The System.Diagnostics namespace provides classes that allow you to interact with system processes, event logs, and performance counters.

Using namespace System::IO : The System.IO namespace provides read access . Opening and reading the files for read access is an important part of IO functionality .This is useful for reading text files but does not work for binary files .

This component is designed for a Windows Forms environment.

Database function's:

OleDb::OleDbConnection : Represents an open connection to a data source.

- Namespace: System.Data.OleDb
- Assembly: System.Data.dll

ConnectionSting : is an easy-to-use reference of connection
strings for numerous databases and data stores.

OleDbConnection(connectingstring) : Gets or sets the string used to open a database.

- Namespace: System.Data.OleDb
- Assembly: System.Data.dll

OleDbCommand(query, conn) : Executes an Access against the Connection and returns the number of rows affected.

- Namespace: System.Data.OleDb
- Assembly: System.Data.dll

OleDbDataReader : Provides a way of reading a forward-only stream of data rows from a data source. This class cannot be inherited.

- Namespace: System.Data.OleDb
- Assembly: System.Data.dll

Command->ExecuteScalar() : The ExecuteScalar() in C++ SqlCommand Object is using for retrieve a single value from Database after the execution of the SQL Statement. The ExecuteScalar() executes SQL statements as well as Stored Procedure and returned a scalar value on first column of first row in the returned Result Set.

- Namespace: System.Data.OleDb
- Assembly: System.Data.dll

Command->ExecuteNonQuery() : ExecuteNonQuery used for executing queries that does not return any data. It is used to execute the sql statements like update, insert, delete etc. ExecuteNonQuery executes the command and returns the number of rows affected.

Conn.Close() : This will close the connection to the Database.

Major Data Objects:

The following data modules will be presented and managed by the system:

User Authentication / Insertion Module: This object will manage connections to the Users Database and retrieve appropriate information depending on the user login credentials. It will also contain processing to add a new user to specific group in the database.

Data Entry Module : This module will encompass all the data entry to be done in the system.

Data Read Module : this module will read the scores/progress of each of the student and translate that into a graph.

Potential users include primary students, Guardians, Superuser.

Login Page

Default Display of Login Page :

- UserNametxt->Text=""; Used to initialise column of username as empty.
- Passwordtxt->Text=""; Used to initialise column of password as empty.
- answer->Text=""; Used to initialise column of answer as empty.
- question->SelectedIndex = -1; Used to initialise column of selected question as -1.
- NewPass->Text=""; Used to initialise column of new password as empty.
- Passwordtxt->PasswordChar='*'; Used for hiding text in password column with "*".
- groupBox1->Visible=false; Initialised to be invisible and will be visible if and only if he/she clicks on security button.

con->ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=FunBrainzForKids.accdb;"; Used to initialise a string named con to establish connection with database named FunBrainzForKids.

Void cleartext() : This function is used to clear texts in every columns so in this it is initiated as default display of login page along with showpass->Checked=false;

- showpass->Checked=false;

String ^ updatetext(String ^ s) : This function is used to remove the spaces at start and end that entered in an input text.

- s = s->TrimEnd(' '): Used to remove spaces at the end of string.
- s = s->TrimStart(' ') : Used to remove spaces in starting of the string.
- return s : Used to return the string as the function is a string type.

Instructions for Login Button : This will check whether the entered username and password are matched in the database and if they are matched it will redirect to homepage else it will display a message showing either username or password are wrong.

Void cleartext() : This function is used to clear texts in every columns so in this it is initiated as default display of login page along with showpass->Checked=false;

String ^ SuggestPassword() : This function creates a random string of size 20 which are choosen randomly from all letters and symbols. For this we have used rand() function which is included in library function.

int send(String ^ email) : This function will check for the email of corresponding user and will generate random password using SuggestPassword() function and that password will be sent to user's email.

Void update() : This function will take the input and update the password in the database.

int SearchUserName() : This function will check whether username exists or not in the database and if exists it will return '1' else '0'.

Void Forget_Click(System::Object^ sender, System::EventArgs^ e) : On clicking this button it will check for the entered username in the database then it will send mail using send(String ^ email) function to corresponding mail of the user.

Security Question Button : On clicking this button this will make group Box1 visible which contains options for security question and column to type its answer and column for new password.

Boolean Passwordvalidate(String ^ s) : This function will check the validity of entered text whether it is non-empty and whether its length lies in between "6-15".

Void Submitbtn_Click(System::Object^ sender, System::EventArgs^ e) : On clicking this button it will check whether the selected question and answer matched in database if so it will update the password using update() function.

SignUp button : On clicking this button it will redirect to signup form.

Sign Up Form

This form contains columns various fields like username, password, first-name, class, email, contact number etc.

Functions:

Boolean uservalidate (String ^ s)

Boolean userdatavalidate (String ^ s)

Boolean classvalidate (String ^ s)

Boolean Mobilevalidate (String ^ s)

Boolean Emailvalidate (String ^ s)

Boolean namevalidate(String ^ s)

Boolean gendervalidate(String ^ s)

Boolean Schoolvalidate(String ^ s)

Boolean Questionvalidate(String ^ s)

Boolean Answervalidate(String ^ s)

Boolean Passwordvalidate(String ^ s)

String ^ updatetext(String ^ s)

Boolean datavalid()

Signupbtn_Click : On clicking this button it will check all the validations through datavalid() function and if all the entered strings are valid then it will be entered into database.

Back Button: This will signup form and redirects to login form.

HomePage

This form contains two panels and 7 buttons.

Buttons :

Profile: On clicking this button it will display “profile.h” in the panel 1.

Graph Button: On clicking this button it will display “arithmetic1.h” which contains a chart in panel1 to show performance graph.

Basic Operation: On clicking this button it will display options to enter three games namely cricket, fill operators, cloud in the panel2.

Abacus: On clicking this Button it will redirect to “Abacus.h”.

Identifying Shapes: On clicking this button it will redirect to “shapesPanel.h” form.

Quizzes: On clicking this button it will display options to enter three games namely 2048, pipeline game, image quiz in panel2 in which they corresponds to "Game_2048.h", "Level_pipeline.h", "Image_Quiz.h" respectively.

Delete account: On clicking this button it will delete the user account from database.

Back Button: On clicking this button it will close homepage form and opens login form.

Profile

This form contains various fields like username, password, first-name, last-name etc.

In these columns it will display corresponding details of the user.

It contains edit button, save button, upload profile pic button, delete profile pic button.

Buttons

Edit Button: On clicking this button it will show an option to edit his details in various fields as displayed.

Save Button: On clicking this button it will check the validity of the strings through datavalid() function (similar function as in signup form) in respective columns and if they are valid it will update new details in the various columns in database.

Upload Image: On clicking this button it will browse the computer and provides option to select an image of various formats like jpg, jpeg, gif etc. After selecting image it will save as <username>.png format.

Delete Image: On clicking this button it will clear the present image and will display a default image which seems to be an empty profile pic.

Quiz’s and Puzzles Module:

There are three main games or puzzles in this module which will help a student to increase his/her IQ and sharpening one’s brain enough to solve complicated puzzles.

This module has three sub modules:

- 2048 Game
- Image recognition
- Pipeline Game **(Backtracking)**

2048 Game:

The game's objective is to slide numbered tiles on a grid to combine them to create a tile with the number 2048. *2048* is played on a gray 4×4 grid, with numbered tiles that slide smoothly when a player moves them using the four arrow keys.

Every turn, a new tile will randomly appear in an empty spot on the board with a value of either 2 or 4.Tiles slide as far as possible in the chosen direction until they are stopped by either another tile or the edge of the grid. If two tiles of the same number collide while moving, they will merge into a tile with the total value of the two tiles that collided. The resulting tile cannot merge with another tile again in the same move. Higher-scoring tiles emit a soft glow.

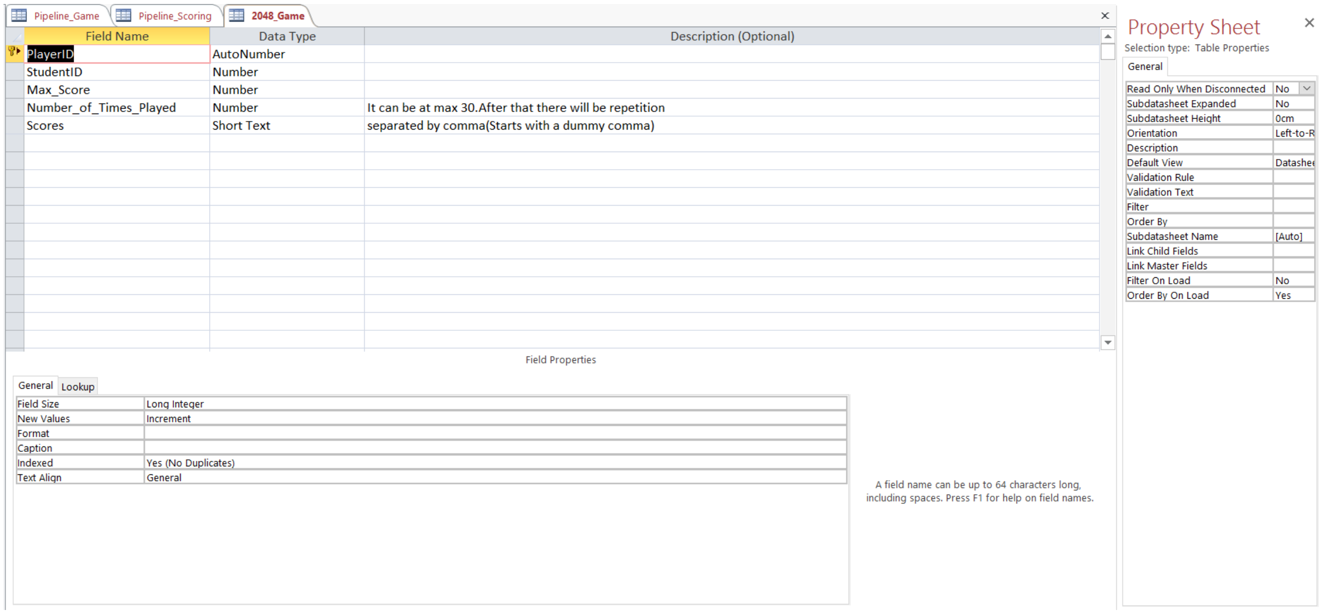
Algorithm:

```
- walk over the array from the first to the last number
  - for each original number in the array that is not zero
    - look backwards for a target position that does not contain a zero (unless it is position zero)
      - if the target position does not contain the original number use the next position
    - if the target position is different from the original position
      - add the number to the number on the target position
      - replace the original number by zero
```

If all slider that are: left, Right, Up, Down are disabled then the game is over and the score is updated.

If at any slide user is able to form a new big tile then the score is incremented by that newly merged nuber.

<u>Database:</u>



Field Name	Data Type	Description (Optional)
PlayerID	AutoNumber	
StudentID	Number	
Max_Score	Number	
Number_of_Times_Played	Number	It can be at max 30.After that there will be repetition
Scores	Short Text	separated by comma(Starts with a dummy comma)

Field Properties

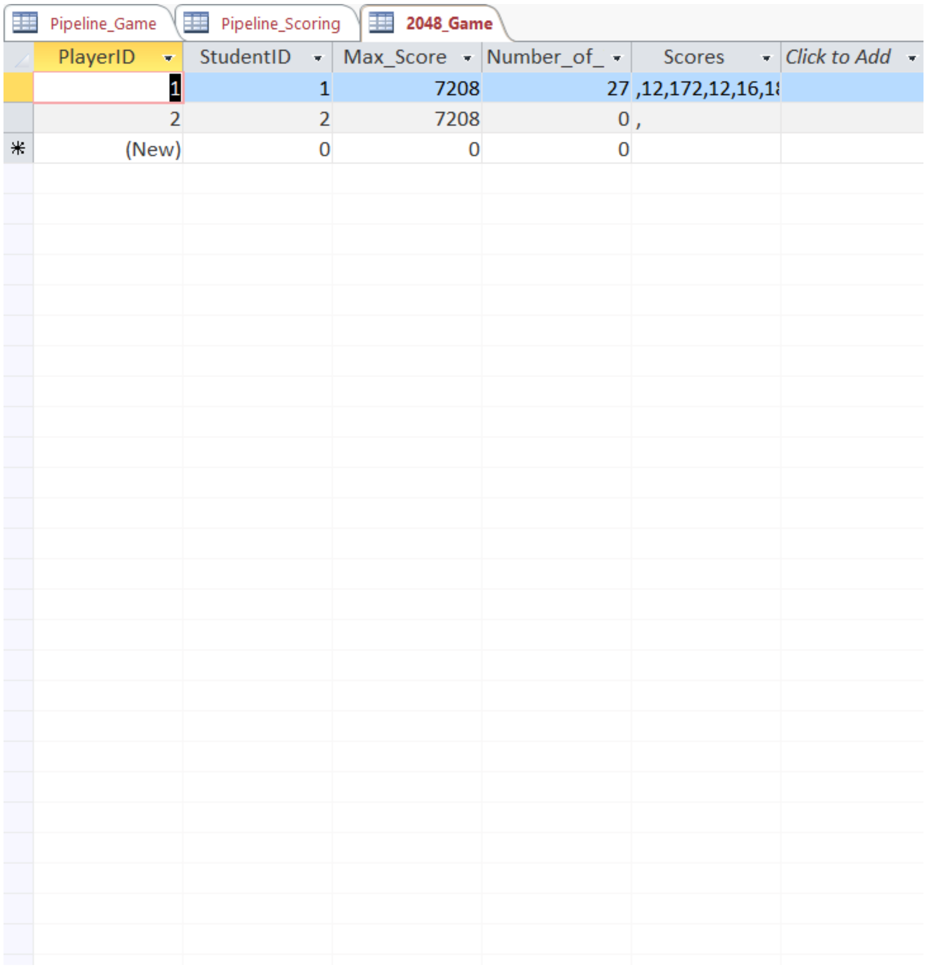
Field Size	Long Integer
New Values	Increment
Format	
Caption	
Indexed	Yes (No Duplicates)
Text Align	General

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

Property Sheet

Selection type: Table Properties

Read Only When Disconnected	No
Subdatasheet Expanded	No
Subdatasheet Height	0cm
Orientation	Left-to-R
Description	
Default View	Datashee
Validation Rule	
Validation Text	
Filter	
Order By	
Subdatasheet Name	[Auto]
Link Child Fields	
Link Master Fields	
Filter On Load	No
Order By On Load	Yes



PlayerID	StudentID	Max_Score	Number_of_	Scores	Click to Add
1	1	7208	27	,12,172,12,16,1	
2	2	7208	0	,	
(New)	0	0	0		

Maximum score is the maximum score the student has scored throughout his gameplay, and the scores are stored as comma separated 30 scores after which if a new score is added the very first score is removed and the new one isupdated in the last.

Functions:

void ColorChanger(array<Button ^, 2> ^buttons) : This is used to change the color of tiles having different number. Each bigger number is represented by different intensities of orange color.

static_cast<System::Int32> : Converts an *expression* to the type of *type-id,* based only on the types that are present in the expression.

convert_string_to_int() : This is an easy way to convert strings of digits into ints, floats or doubles. Following is a sample program using a stringstream to convert string to int.

bool isFull() : Function to check whether the grid is full or not.

button->Enabled: To check whether the button is Enabled or not.

rand(): rand() function is used in C++ to generate random numbers. If we generate a sequence of random number with rand() function, it will create the same sequence again and again every time program runs.

bool is_Game_Over() : Function which checks whether any of the slider works or not.

If any slider slide's the grid to any direction then game is continued else game is over.

void Move_Right() : Function used for sliding the grid to right and merging if possible.

Same for up, left and down.

MessageBox.Show() : Will pop up a window having some developer entered information/Instruction.

void is_2048_found() : This function will check all the tile. If any one of tile is containing 2048, It's the win condition hence a message box will popout displaying "You Won!". And all the buttons are then Disabled.

In this game keypress is also noticed, If a user instead of hitting the buttons press's he direction key from the keyboard then also the game works.

Image Recognition:

This is similar to the logo quiz available in the play-store. In this module initially 9 objects are defined which can be extended to whatever number one want. Technically we have implemented a code which searches for the object folder and reads the images then intelligently generate that number of boxes which are required to specify the object’s reference/Name. This will provide 14 random alphabets as options from which student has to identify what is the reference of the image. Those 14 random alphabets will surely contain alphabets required to complete the name of the object.

Levels are specified as word size:

- Word size 1-4 : Easy
- Word size 5-8 : Medium
- Word size above 8 : Hard

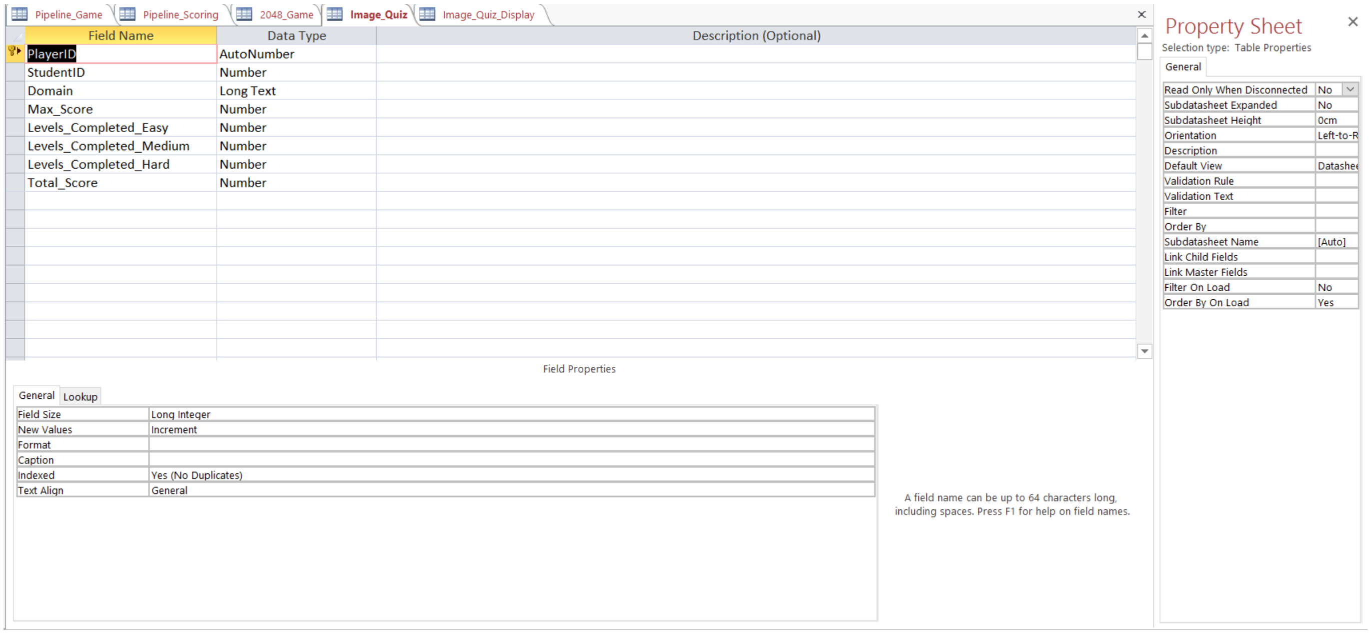
Hints:

Hints are of 4 type :

- Random alphabet : Will provide a random correct alphabet placed in its correct position. (score deducted
- Place selector: User can select at which place’s he/she want the hint.
- Complete word : Will complete the whole answer.

(score deducted as mentioned in the respective hint box).

Database:



Field Name	Data Type	Description (Optional)
PlayerID	AutoNumber	
StudentID	Number	
Domain	Long Text	
Max_Score	Number	
Levels_Completed_Easy	Number	
Levels_Completed_Medium	Number	
Levels_Completed_Hard	Number	
Total_Score	Number	

Field Properties

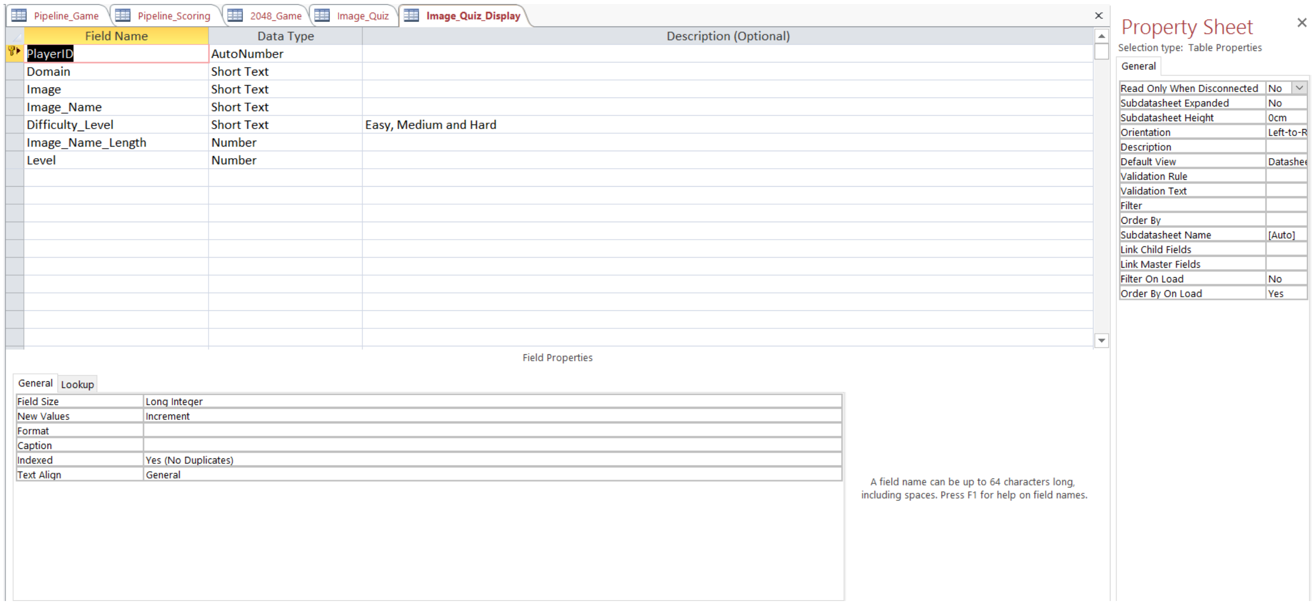
Field Size	Long Integer
New Values	Increment
Format	
Caption	
Indexed	Yes (No Duplicates)
Text Align	General

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

Property Sheet

Selection type: Table Properties

Read Only When Disconnected	No
Subdatasheet Expanded	No
Subdatasheet Height	0cm
Orientation	Left-to-R
Description	
Default View	Datashee
Validation Rule	
Validation Text	
Filter	
Order By	
Subdatasheet Name	[Auto]
Link Child Fields	
Link Master Fields	
Filter On Load	No
Order By On Load	Yes



Field Name	Data Type	Description (Optional)
PlayerID	AutoNumber	
Domain	Short Text	
Image	Short Text	
Image_Name	Short Text	
Difficulty_Level	Short Text	Easy, Medium and Hard
Image_Name_Length	Number	
Level	Number	

Field Properties

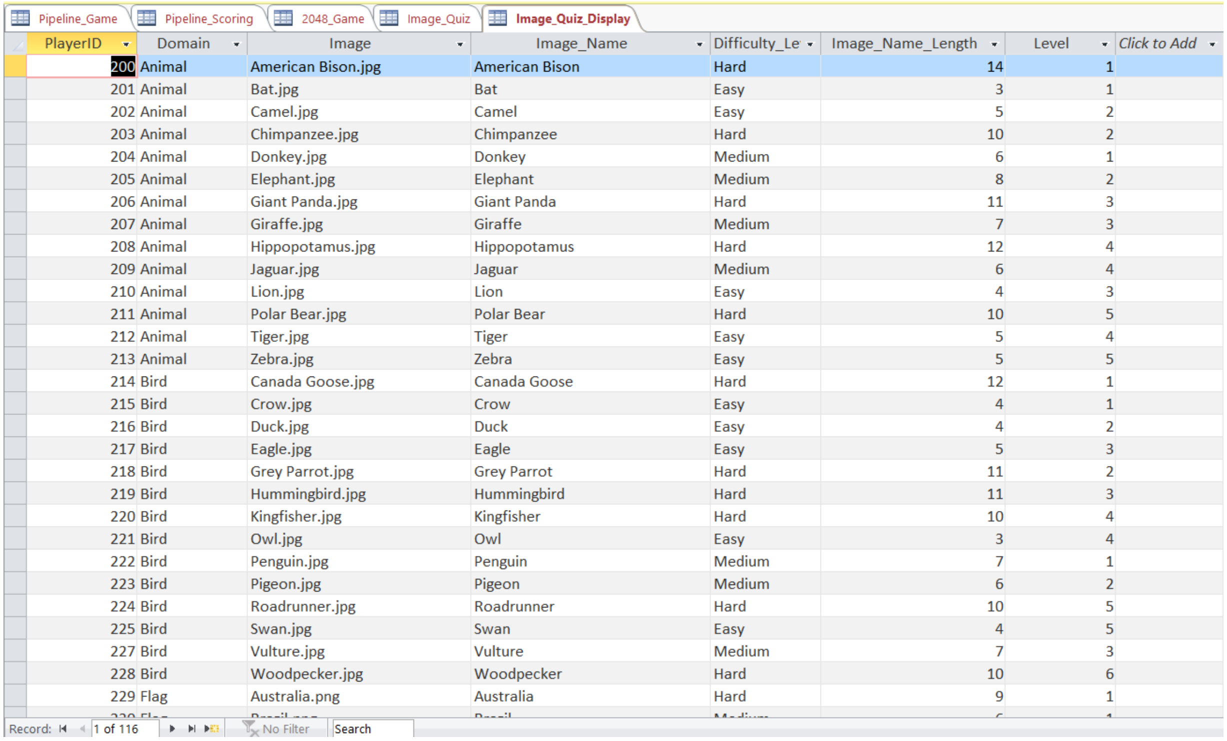
Field Size	Long Integer
New Values	Increment
Format	
Caption	
Indexed	Yes (No Duplicates)
Text Align	General

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

Property Sheet

Selection type: Table Properties

Read Only When Disconnected	No
Subdatasheet Expanded	No
Subdatasheet Height	0cm
Orientation	Left-to-R
Description	
Default View	Datashee
Validation Rule	
Validation Text	
Filter	
Order By	
Subdatasheet Name	[Auto]
Link Child Fields	
Link Master Fields	
Filter On Load	No
Order By On Load	Yes



PlayerID	Domain	Image	Image_Name	Difficulty_Le	Image_Name_Length	Level	Click to Add
200	Animal	American Bison.jpg	American Bison	Hard	14	1	
201	Animal	Bat.jpg	Bat	Easy	3	1	
202	Animal	Camel.jpg	Camel	Easy	5	2	
203	Animal	Chimpanzee.jpg	Chimpanzee	Hard	10	2	
204	Animal	Donkey.jpg	Donkey	Medium	6	1	
205	Animal	Elephant.jpg	Elephant	Medium	8	2	
206	Animal	Giant Panda.jpg	Giant Panda	Hard	11	3	
207	Animal	Giraffe.jpg	Giraffe	Medium	7	3	
208	Animal	Hippopotamus.jpg	Hippopotamus	Hard	12	4	
209	Animal	Jaguar.jpg	Jaguar	Medium	6	4	
210	Animal	Lion.jpg	Lion	Easy	4	3	
211	Animal	Polar Bear.jpg	Polar Bear	Hard	10	5	
212	Animal	Tiger.jpg	Tiger	Easy	5	4	
213	Animal	Zebra.jpg	Zebra	Easy	5	5	
214	Bird	Canada Goose.jpg	Canada Goose	Hard	12	1	
215	Bird	Crow.jpg	Crow	Easy	4	1	
216	Bird	Duck.jpg	Duck	Easy	4	2	
217	Bird	Eagle.jpg	Eagle	Easy	5	3	
218	Bird	Grey Parrot.jpg	Grey Parrot	Hard	11	2	
219	Bird	Hummingbird.jpg	Hummingbird	Hard	11	3	
220	Bird	Kingfisher.jpg	Kingfisher	Hard	10	4	
221	Bird	Owl.jpg	Owl	Easy	3	4	
222	Bird	Penguin.jpg	Penguin	Medium	7	1	
223	Bird	Pigeon.jpg	Pigeon	Medium	6	2	
224	Bird	Roadrunner.jpg	Roadrunner	Hard	10	5	
225	Bird	Swan.jpg	Swan	Easy	4	5	
227	Bird	Vulture.jpg	Vulture	Medium	7	3	
228	Bird	Woodpecker.jpg	Woodpecker	Hard	10	6	
229	Flag	Australia.png	Australia	Hard	9	1	

Sample database storing the data as domain, image, image name(required for dynamically plotting the spaces for the answer provided by user).

Functions:

void read_directory(string folder) : Function to read the complete directory where argument passed is the name of the object chosen by the student.

Debug::WriteLine() : Writes information about the debug to the trace listeners in the Listeners collection.

- Namespace : System.Diagnostics
- Assemblies : System.Diagnostics.Debug.dll, System.dll, netstandard.dll

FindFirstFile() : Searches a directory for a file or subdirectory with a name that matches a specific name (or partial name if wildcards are used). To specify additional attributes to use in a search, use the FindFirstFileEx function. To perform this operation as a transacted operation, use the FindFirstFileTransacted function.

private: System::Void ImageQuiz2_Load_1(System::Object^ sender, System::EventArgs^ e) : Form load function, what are the things which will run at the form load event.

Pipeline:

This module have a 5X5 grid in which one have to make a path from “Top-left” tiles’s center to the center of the “Right-Bottom” tile. The grid is having three type of pipes named as A, B, C. The A types have a straight pipe connecting the centers to two opposite sides of the square, the B are Quarter in shape and C type for the “top left” and “Right bottom” tile the half of the A.

We implemented Backracking here as many paths are possible and according to our result and condition we have 8152 different number of paths.

On clicking the tile The tile will rotate clockwise according to it’s type if A then the first one is Horizontal and other one Vertical to maintain the modularity we have given numbering from 0-3 where 0,2 are horizontal and 1,3 are vertical representation respectively. And B – all different as four types of representation is possible.

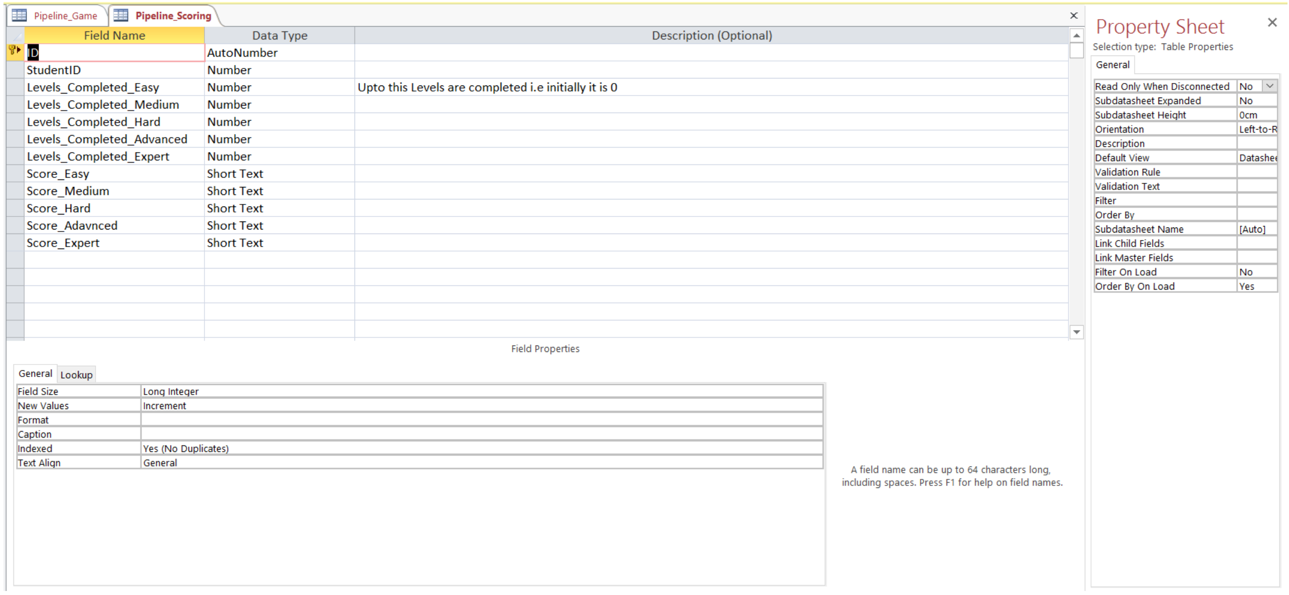
The paths are represented as sequence of R, D, U and L representing the directions of movement. The Answer is stored in the database as sequence of integers using 0, 1, 2, 3 and 5. where 5 is for garbage.

When the user makes a path then generated integer string is then compared with all the answer available in the database i.e. 8152 so to remove the misunderstanding. Suppose one make the path and that answer is not stored for the given question in the database then program will show wrong message whereas that answer is correct.

Levels:

Total 8152 possible path so to distribute that to 5 different levels we just used simple mathematics to divide and we got around ‘34’ games for each level .

Database:

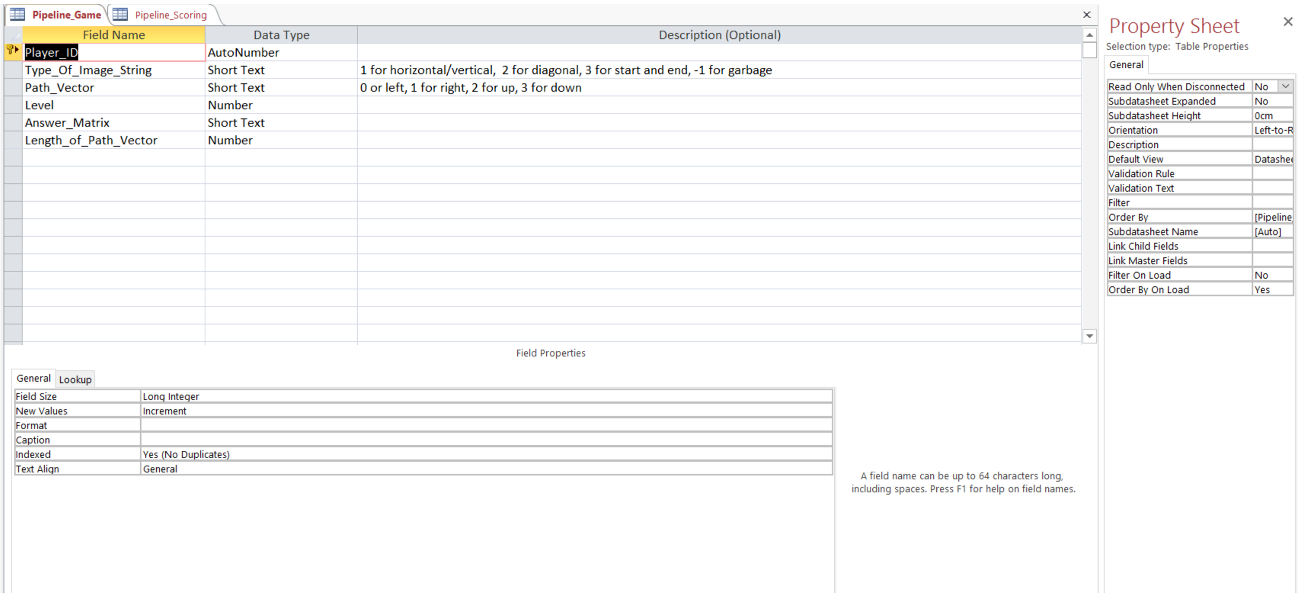


Field Name	Data Type	Description (Optional)
ID	AutoNumber	
StudentID	Number	
Levels_Completed_Easy	Number	Upto this Levels are completed i.e initially it is 0
Levels_Completed_Medium	Number	
Levels_Completed_Hard	Number	
Levels_Completed_Advanced	Number	
Levels_Completed_Expert	Number	
Score_Easy	Short Text	
Score_Medium	Short Text	
Score_Hard	Short Text	
Score_Adavnced	Short Text	
Score_Expert	Short Text	

Field Size	Long Integer
New Values	Increment
Format	
Caption	
Indexed	Yes (No Duplicates)
Text Align	General

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

Read Only When Disconnected	No
Subdatasheet Expanded	No
Subdatasheet Height	0cm
Orientation	Left-to-R
Description	
Default View	Datashe
Validation Rule	
Validation Text	
Filter	
Order By	
Subdatasheet Name	[Auto]
Link Child Fields	
Link Master Fields	
Filter On Load	No
Order By On Load	Yes

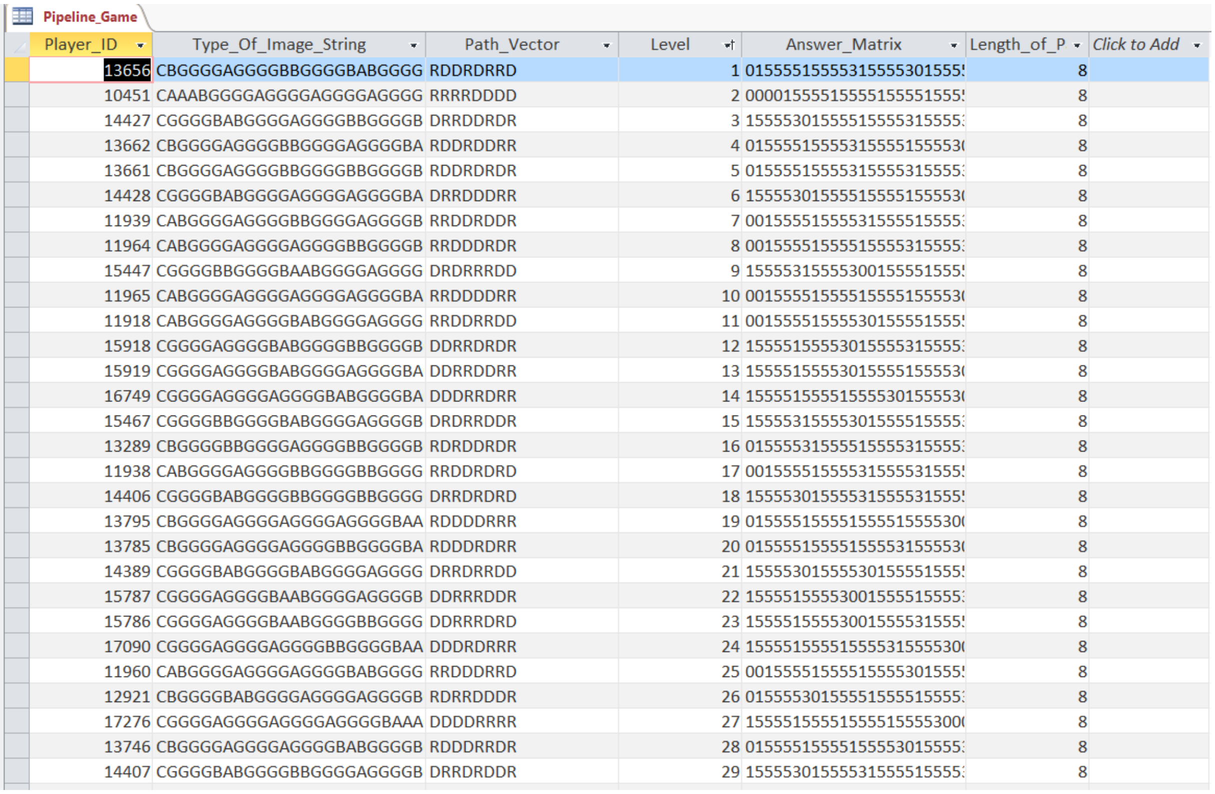


Field Name	Data Type	Description (Optional)
Player_ID	AutoNumber	
Type_Of_Image_String	Short Text	1 for horizontal/vertical, 2 for diagonal, 3 for start and end, -1 for garbage
Path_Vector	Short Text	0 or left, 1 for right, 2 for up, 3 for down
Level	Number	
Answer_Matrix	Short Text	
Length_of_Path_Vector	Number	

Field Size	Long Integer
New Values	Increment
Format	
Caption	
Indexed	Yes (No Duplicates)
Text Align	General

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

Read Only When Disconnected	No
Subdatasheet Expanded	No
Subdatasheet Height	0cm
Orientation	Left-to-R
Description	
Default View	Datashe
Validation Rule	
Validation Text	
Filter	
Order By	[Pipeline
Subdatasheet Name	[Auto]
Link Child Fields	
Link Master Fields	
Filter On Load	No
Order By On Load	Yes



Pipeline_Game

Player_ID	Type_Of_Image_String	Path_Vector	Level	Answer_Matrix	Length_of_P	*Click to Add*
13656	CBGGGGAGGGGBBGGGGBABGGGG	RDDRDRRD	1	015555155553155553015555	8	
10451	CAAABGGGGAGGGGAGGGGAGGGG	RRRRDDDD	2	000015555155551555515555	8	
14427	CGGGGBABGGGGAGGGGBBGGGGB	DRRDDRDR	3	155553015555155553155553	8	
13662	CBGGGGAGGGGBBGGGGAGGGGBA	RDDRDDRR	4	015555155553155551555530	8	
13661	CBGGGGAGGGGBBGGGGBBGGGGB	RDDRDRDR	5	015555155553155553155553	8	
14428	CGGGGBABGGGGAGGGGAGGGGBA	DRRDDDRR	6	155553015555155551555530	8	
11939	CABGGGGAGGGGBBGGGGAGGGGB	RRDDRDDR	7	001555515555315555155553	8	
11964	CABGGGGAGGGGAGGGGBBGGGGB	RRDDDRDR	8	001555515555155553155553	8	
15447	CGGGGBBGGGGBAABGGGGAGGGG	DRDRRRDD	9	155553155553001555515555	8	
11965	CABGGGGAGGGGAGGGGAGGGGBA	RRDDDDRR	10	001555515555155551555530	8	
11918	CABGGGGAGGGGBABGGGGAGGGG	RRDDRRDD	11	001555515555301555515555	8	
15918	CGGGGAGGGGBABGGGGBBGGGGB	DDRRDRDR	12	155551555530155553155553	8	
15919	CGGGGAGGGGBABGGGGAGGGGBA	DDRRDDRR	13	155551555530155551555530	8	
16749	CGGGGAGGGGAGGGGBABGGGGBA	DDDRRDRR	14	155551555515555301555530	8	
15467	CGGGGBBGGGGBABGGGGAGGGGB	DRDRRDDR	15	155553155553015555155553	8	
13289	CBGGGGBBGGGGAGGGGBBGGGGB	RDRDDRDR	16	015555315555155553155553	8	
11938	CABGGGGAGGGGBBGGGGBBGGGG	RRDDRDRD	17	001555515555315555315555	8	
14406	CGGGGBABGGGGBBGGGGBBGGGG	DRRDRDRD	18	155553015555315555315555	8	
13795	CBGGGGAGGGGAGGGGAGGGGBAA	RDDDDRRR	19	015555155551555515555300	8	
13785	CBGGGGAGGGGAGGGGBBGGGGBA	RDDDRDRR	20	015555155551555531555530	8	
14389	CGGGGBABGGGGBABGGGGAGGGG	DRRDRRDD	21	155553015555301555515555	8	
15787	CGGGGAGGGGBAABGGGGAGGGGB	DDRRRDDR	22	155551555530015555155553	8	
15786	CGGGGAGGGGBAABGGGGBBGGGG	DDRRRDRD	23	155551555530015555315555	8	
17090	CGGGGAGGGGAGGGGBBGGGGBAA	DDDRDRRR	24	155551555515555315555300	8	
11960	CABGGGGAGGGGAGGGGBABGGGG	RRDDDRRD	25	001555515555155553015555	8	
12921	CBGGGGBABGGGGAGGGGAGGGGB	RDRRDDDR	26	015555301555515555155553	8	
17276	CGGGGAGGGGAGGGGAGGGGBAAA	DDDDRRRR	27	155551555515555155553000	8	
13746	CBGGGGAGGGGAGGGGBABGGGGB	RDDDRRDR	28	015555155551555530155553	8	
14407	CGGGGBABGGGGBBGGGGAGGGGB	DRRDRDDR	29	155553015555315555155553	8	

Database containing the whole answers for the grid [C, B, G, A are the type of imageswhich thetile contains stored in row wise format].

Functions:

void making_encodings(int n) : Function which generates the path_vector using the direction. L=0, R=1, U=2, D=3, Garbage=5.

Int ** = pointer to pointer/ Used for 2-Dimensional array.

private: System::Void pictureBox_Click(System::Object^ sender, System::EventArgs^ e) : Function to rotate the image clockwise to which the usr has clicked.

System::Int64::Parse() : Converts the string representation of a number to its 64-bit signed integer equivalent.

- Namespace: System
- Assemblies: System.Runtime.dll, mscorlib.dll, netstandard.dll

private: System::Void btn_Back_To_Main_From_Pipeline_Game_Click(System::Object^ sender, System::EventArgs^ e) : Function for the back button. It will take user to pipeline front form.

private: System::Void timer_Tick(System::Object^ sender, System::EventArgs^ e) : Function for the timer clock, Stars are dependent of the time left.

private: System::Void button_Click(System::Object^ sender, System::EventArgs^ e) : button click event.

IsSolved : an Integer vriable which is used to check whether the puzzle is solved or not by matching the string generated by the user with the total answer strings available in the database.

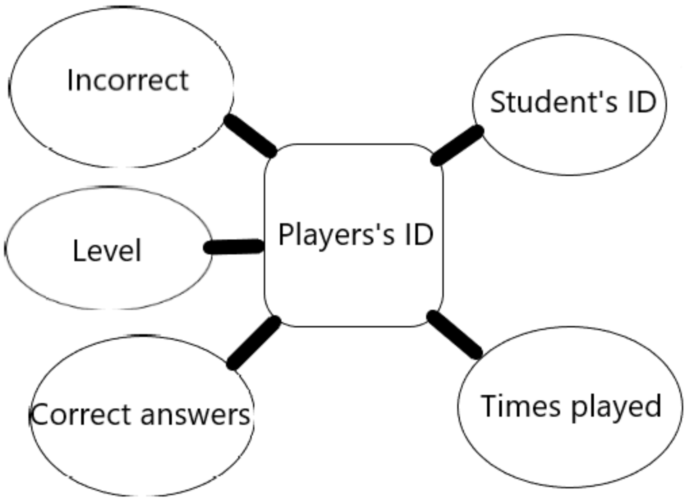
Arithmetic Module:

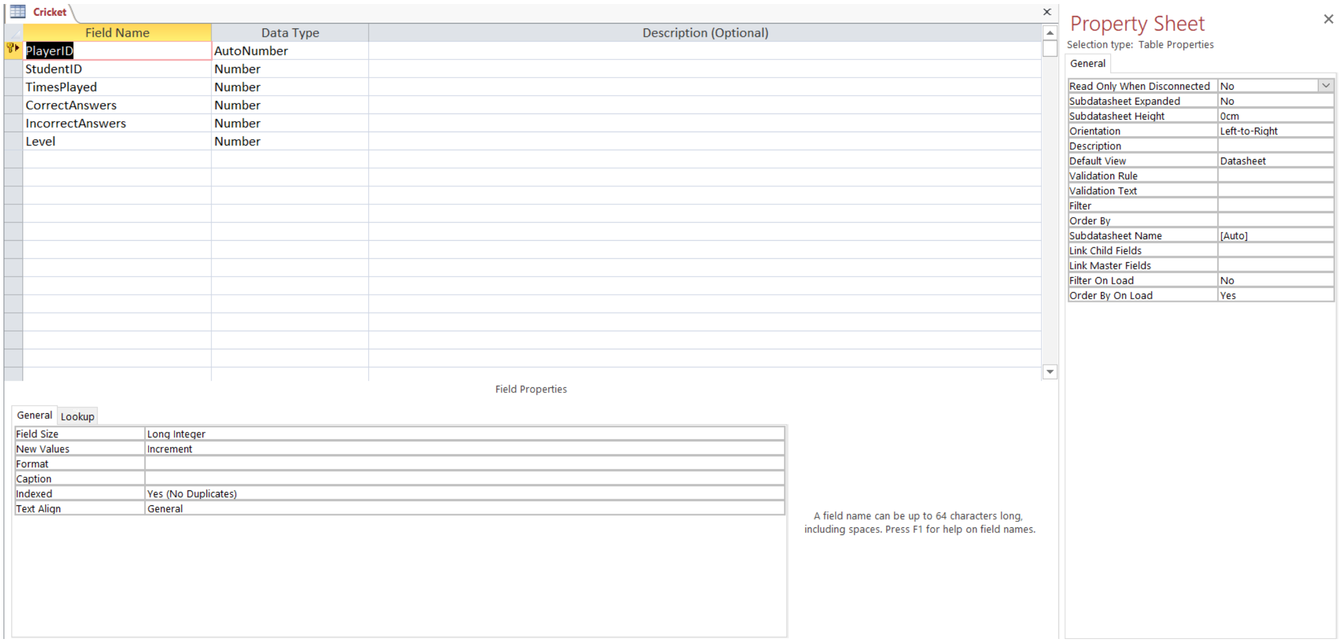
The purpose of this Module is to teach a primary student, the basic of Addition, Subtraction, Multiplication and Division. This application comprises of various Quiz's and Puzzles helping a student to apply his brain and implement the concepts well meanwhile enjoying the Game's too. The student first has to register, to use the services and the credentials will be handled by his/her guardians, so that they can keep track on his/her daily usage and his overall progress.

We will provide a graph showing the progress of a student according to his performance in all the Quiz's and Puzzles by keeping track on his record. We are focusing on progress as well as the time spend on this application each day.

Database:

Cricket:





Field Name	Data Type	Description (Optional)
PlayerID	AutoNumber	
StudentID	Number	
TimesPlayed	Number	
CorrectAnswers	Number	
IncorrectAnswers	Number	
Level	Number	

Field Properties

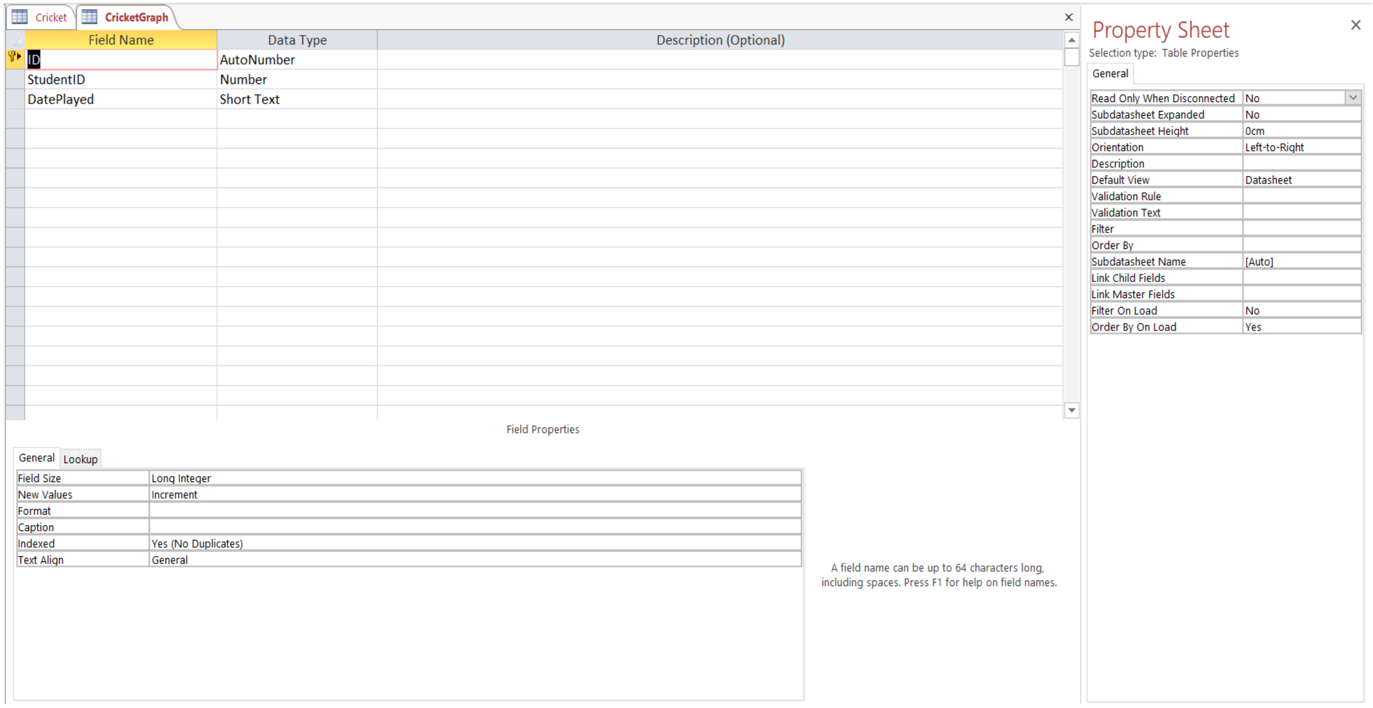
Field Size	Long Integer
New Values	Increment
Format	
Caption	
Indexed	Yes (No Duplicates)
Text Align	General

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

Property Sheet

Selection type: Table Properties

Read Only When Disconnected	No
Subdatasheet Expanded	No
Subdatasheet Height	0cm
Orientation	Left-to-Right
Description	
Default View	Datasheet
Validation Rule	
Validation Text	
Filter	
Order By	
Subdatasheet Name	[Auto]
Link Child Fields	
Link Master Fields	
Filter On Load	No
Order By On Load	Yes



Field Name	Data Type	Description (Optional)
ID	AutoNumber	
StudentID	Number	
DatePlayed	Short Text	

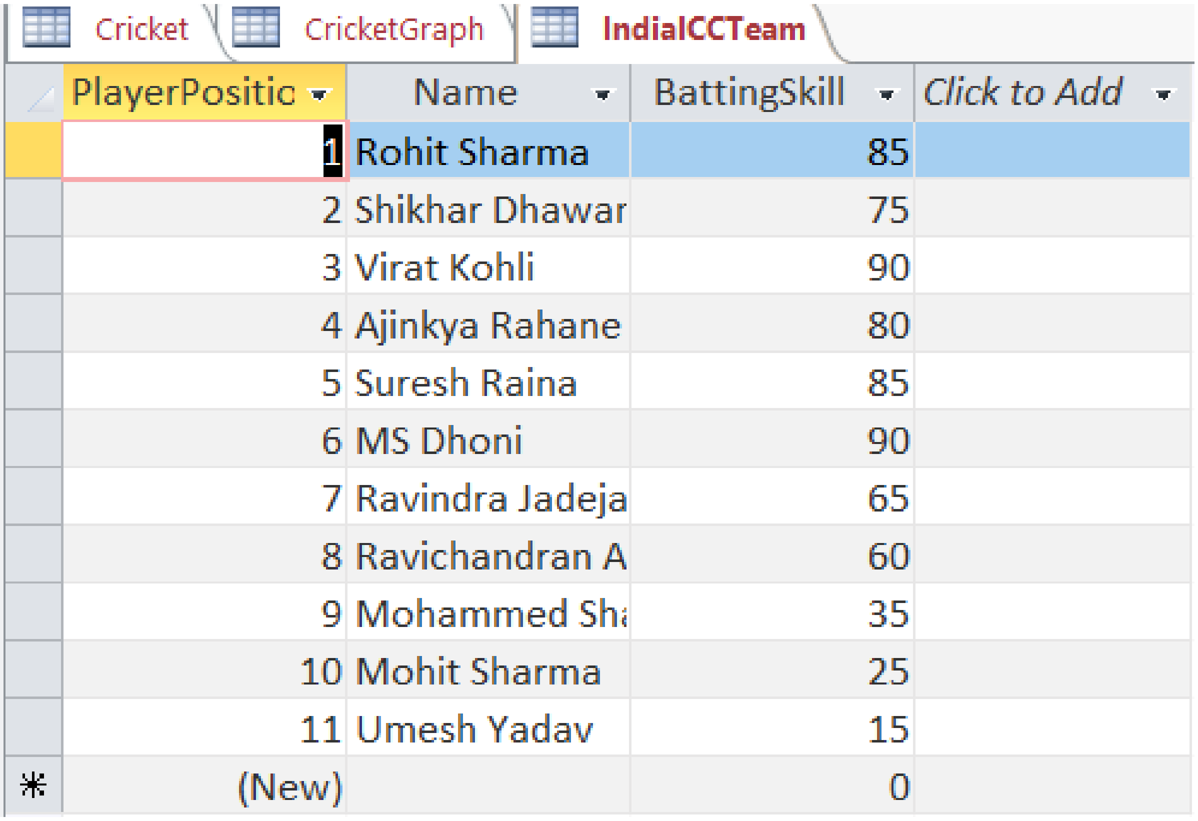
Field Size	Long Integer
New Values	Increment
Format	
Caption	
Indexed	Yes (No Duplicates)
Text Align	General

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

Property Sheet

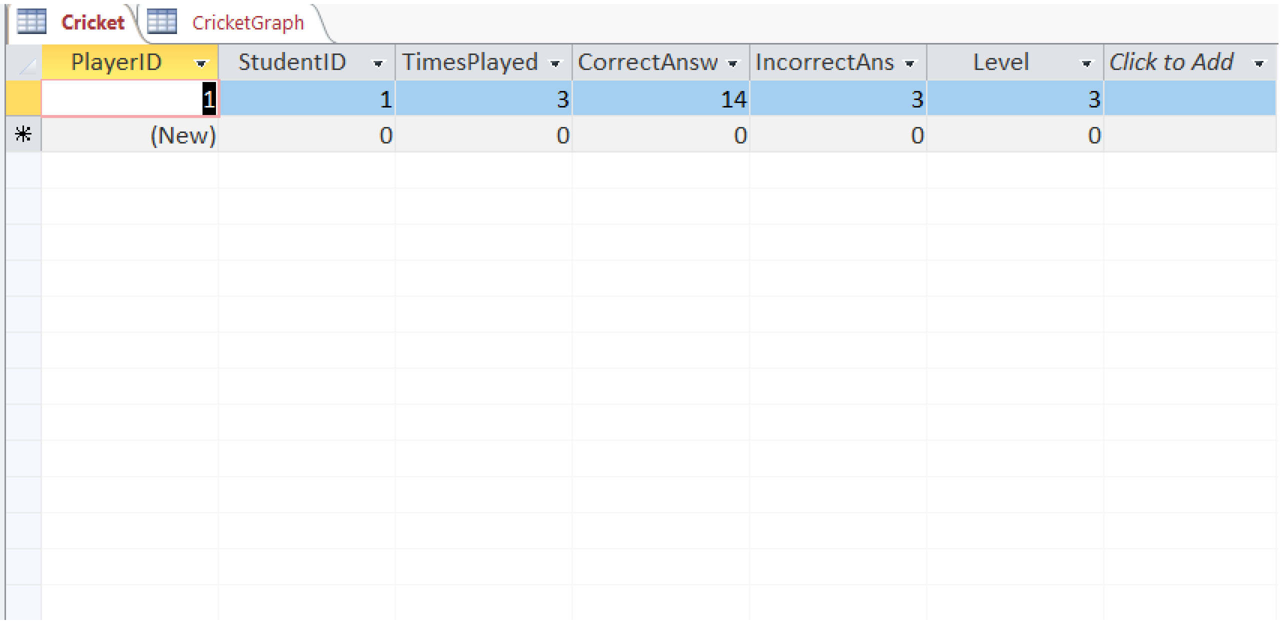
Selection type: Table Properties

Read Only When Disconnected	No
Subdatasheet Expanded	No
Subdatasheet Height	0cm
Orientation	Left-to-Right
Description	
Default View	Datasheet
Validation Rule	
Validation Text	
Filter	
Order By	
Subdatasheet Name	[Auto]
Link Child Fields	
Link Master Fields	
Filter On Load	No
Order By On Load	Yes



PlayerPositic	Name	BattingSkill	Click to Add
1	Rohit Sharma	85	
2	Shikhar Dhawar	75	
3	Virat Kohli	90	
4	Ajinkya Rahane	80	
5	Suresh Raina	85	
6	MS Dhoni	90	
7	Ravindra Jadeja	65	
8	Ravichandran A	60	
9	Mohammed Sha	35	
10	Mohit Sharma	25	
11	Umesh Yadav	15	
(New)		0	

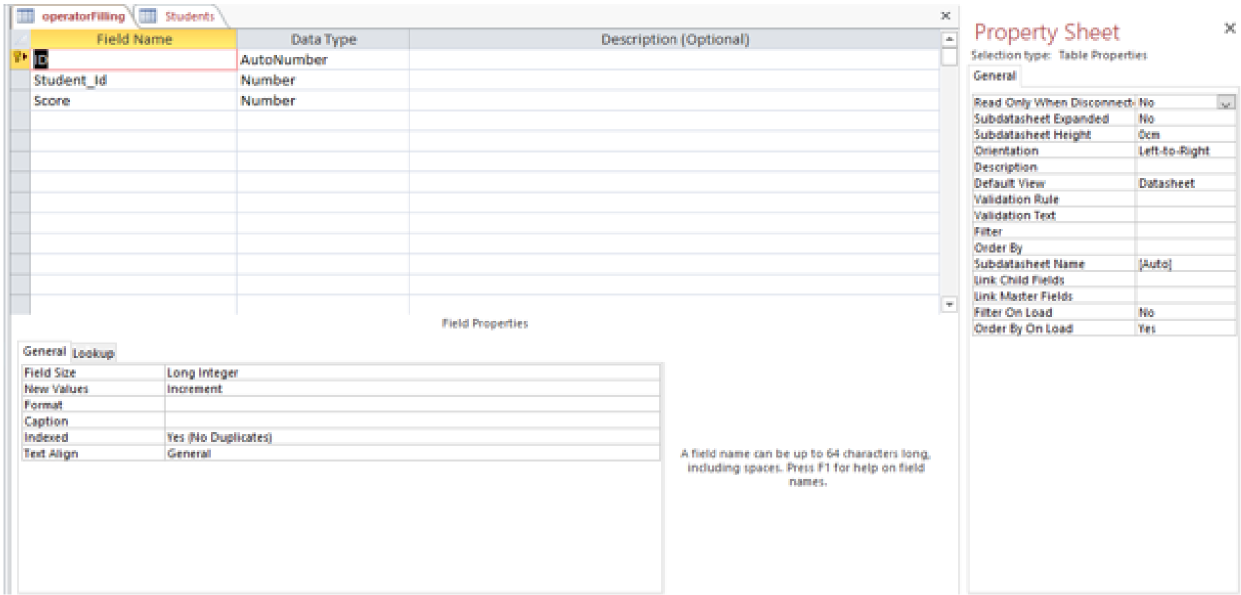
The team database containing player's name and his Batting skills.



PlayerID	StudentID	TimesPlayed	CorrectAnsw	IncorrectAns	Level	Click to Add
1	1	3	14	3	3	
(New)	0	0	0	0	0	

Sample updated database of a Student having StudentId =1. This database is useful for the graph.

Operator_Filling :



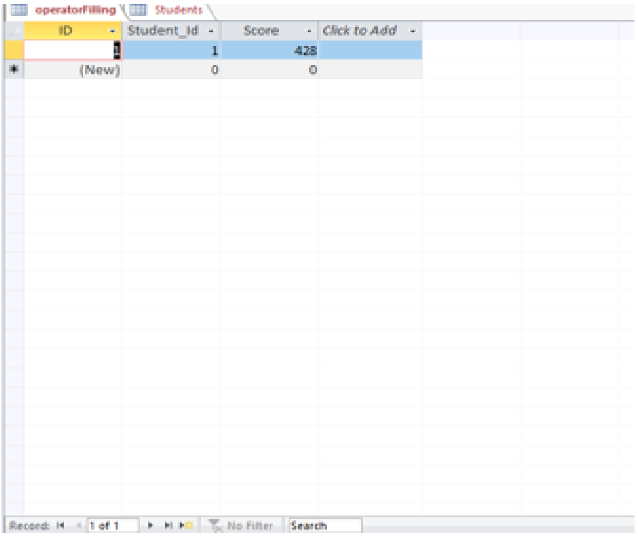
Field Name	Data Type	Description (Optional)
ID	AutoNumber	
Student_Id	Number	
Score	Number	

Field Size	Long Integer
New Values	Increment
Format	
Caption	
Indexed	Yes (No Duplicates)
Text Align	General

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

Selection type: Table Properties

Read Only When Disconnect	No
Subdatasheet Expanded	No
Subdatasheet Height	0cm
Orientation	Left-to-Right
Description	
Default View	Datasheet
Validation Rule	
Validation Text	
Filter	
Order By	
Subdatasheet Name	[Auto]
Link Child Fields	
Link Master Fields	
Filter On Load	No
Order By On Load	Yes



ID	Student_Id	Score	Click to Add
1	1	428	
(New)	0	0	

Functions:

ToString() : We can use toString() method to get string representation of an object. Whenever we try to print the Object reference then internally toString() method is invoked. If we did not define toString() method in your class then Object class toString() method is invoked otherwise our implemented/Overridden toString() method will be called.

MessageBox.Show() : Will pop up a window having some developer entered information/Instruction.

Private:Void QuestionTimer_Tick(System::Object^ sender, System::EventArgs^ e) : You might sometimes want to create a procedure that runs at specific time intervals until a loop has finished or that runs when a set time interval has elapsed. The Timer component makes such a procedure possible.

This component is designed for a Windows Forms environment.

Cricket :

StartGame() : Funtion call to start the game from the beginning.

Private:Void nextQuestion() : Function call to provide new question when answer given to previous question is correct or if wrong answer submitted and he has extra life.

Button->Enabled : Boolean function to check whether button is enabled or not.

std::pair <std::data_tye1, data_type2> Pair_name : The pair container is a simple container defined in <utility> header consisting of two data elements or objects.

- The first element is referenced as ‘first’ and the second element as ‘second’ and the order is fixed (first, second).
- Pair is used to combine together two values which may be different in type. Pair provides a way to store two heterogeneous objects as a single unit.
- Pair can be assigned, copied and compared. The array of objects allocated in a map or hash_map are of type ‘pair’ by default in which all the ‘first’ elements are unique keys associated with their ‘second’ value objects.
- To access the elements, we use variable name followed by dot operator followed by the keyword first or second.

String ^ x = gcnew String(): defining new variable x of string type.

gcnew : The Windows Runtime and common language runtime represent strings as objects whose allocated memory is managed automatically. That is, you are not required to explicitly discard the memory for a string when the string variable goes out of scope or your application ends. To indicate that the lifetime of a string object is to be managed automatically, declare the string type with the handle-to-object (^) modifier.

Textbox->Text = Reads/Writes from/to the Textbox named TextBox.

Label->Text = Reads/Writes from/to the Label named Lable.

setLabels() : Function which initializes all the Labels present in the Form.

Timer->Start() : Starts the timerclock.

Void SliderTimer_Tick(System::Object^ sender, System::EventArgs^ e) : Function to specify what task to be done at a specific period of time defined in the timerclock’s property.

Private:Void evalAnswer(int ans) : Function to evaluate the answer submitted by the student.

Private: System::Void SubmitButton_Click(System::Object^ sender, System::EventArgs^ e) : Function to handle what goes on after clicking the submit button.

Private: System::Void rand_Click(System::Object^ sender, System::EventArgs^ e) : Function to check at which array box the slider is clicked.

Private: System::Void StopButton_Click(System::Object^ sender, System::EventArgs^ e) :

Function to handle what goes on after clicking the stop button int::Parse() : Converts the string representation of a number to its 32-bit signed integer equivalent.

private: Void EndGame() : Function which ends the game, meanwhile updating the progress of the student by updating the scores, time spend, correct answer given, incorrect answer given.

this->Close() : Close's the Cricket Game.

Cloudgame :

#include "GlobalFuncs.h" : This includes the header file GlobalFuncs.h which generate's the random questions of Random or specified operand and level of the question.

#include "cloudgame.h" : This includes the header file cloudgame.h and internally uses all its data.

cloud(Form ^obj) : Constructor passing an Object of form as an argument.

private: System::Void button_Click(System::Object^ sender, System::EventArgs^ e) :

Function which decides which of the category is choosen by the student.

- button1 = Addition.
- Button2 = Subtraction.
- Button3 = Multiplication.
- Button4 = Division.

cloudgame ^ form = gcnew cloudgame(category) : This will create a variable form of cloudgame form type, and call the cloudgame form with category passed as an argument.

form->ShowDialog() : This will popup the Form named as form as mentioned above.

private: System::Void timer_Tick(System::Object^ sender, System::EventArgs^ e) : Funtion to specify what goes on when the timer starts. In this timer is used for the movement of options addition to that it also specifies the speed of the option with which it displaces.

- Timer1 for option1.
- Timer2 for option2.
- Timer3 for option3.
- Timer4 for option4.

private: System::Void cloudgame_Load(System::Object^ sender, System::EventArgs^ e) :

Function to specify what has to be done on the Form Load. We call the function StartGame() in this.

rand() : rand() function is used in C to generate random numbers. If we generate a sequence of random number with rand() function, it will create the same sequence again and again every time program runs.

Textbox->Text = Reads/Writes from/to the Textbox named TextBox.

Label->Text = Reads/Writes from/to the Label named Lable.

private: System::Void button_Click(System::Object^ sender, System::EventArgs^ e) : Function which proceeds when button named button is clicked [here button is the option one of which is having correct answer, Correct leads to next question and incrementing the score else will show the score and correct answer].

this->Close() = Close's the cloudgame button and category form will be active again.

Operator_Filling:

private : System::Void GenerateQuestion (int level) : Function to generate random question depending on the level of the student.

Pow() : Given two numbers base and exponent, pow() function finds x raised to the power of y i.e. x^y .

Syntax:

double pow(double x, double y)**;x :** floating point base value**y :** floating point power value.

BringToFront() : Brings the control to the front of the z-order. The control is moved to the front of the z-order. If the control is a child of another control, the child control is moved to the front of the z-order. BringToFront does not make a control a top-level control, and it does not raise the Paint event.

- Namespace: System.Windows.Forms
- Assembly: System.Windows.Forms.dll

private: System::Void operatorFilling_Load(System::Object^ sender, System::EventArgs^ e) : Function specifying the task's to be done on the form load.

System::Convert::ToString() : Converts the specified value to its equivalent string representation.

- Namespace: System
- Assemblies: System.Runtime.Extensions.dll, mscorlib.dll, netstandard.dll

Timer->Start() : Starts the timer.

private: System::Void submit_Click(System::Object^ sender, System::EventArgs^ e) : Function specifying what goes on after the submit button is clicked.

panel->Hide() : Hides out the panel named as panel.

Textbox->Text = Reads/Writes from/to the Textbox named TextBox.

Label->Text = Reads/Writes from/to the Label named Lable.

ClockTimer->Stop() : This will stop the clock.

private: System::Void ClockTimer_Tick(System::Object^ sender, System::EventArgs^ e) : Function specifying what goes on to the start of the clock.

Identifying Shapes Module:

The purpose of this module is to enhance the visualizing power of the student in the field of identifying different shapes . This module comprises of shape information , draw shapes , a question bank and a exciting game .

Sub-Modules :

Learn Shapes sub-module : This has the information of different shapes along with the picture of that shape for learning purpose .

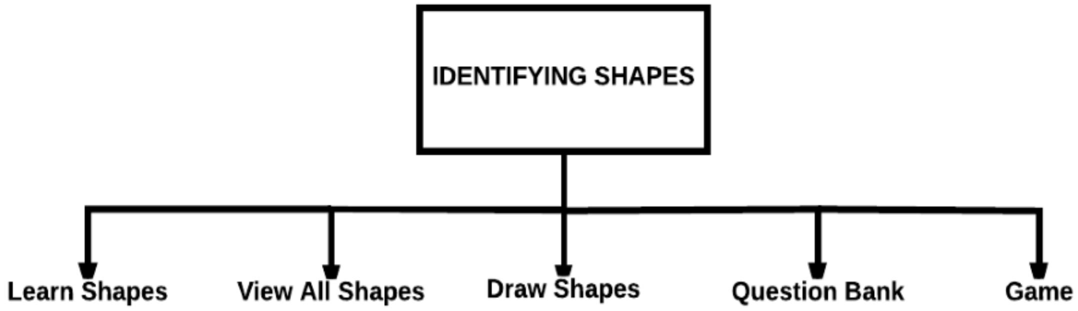
View All Shapes sub-module : In this sub-module , pictures of all the shapes that are included in the database are displayed .By clicking on any shape he/she can go to the information page.

Draw Shapes sub-module : This part is provided with some points joining which user can draw various different types of shapes and can cross check the name of that shape whether it matches with the one which he thought of .

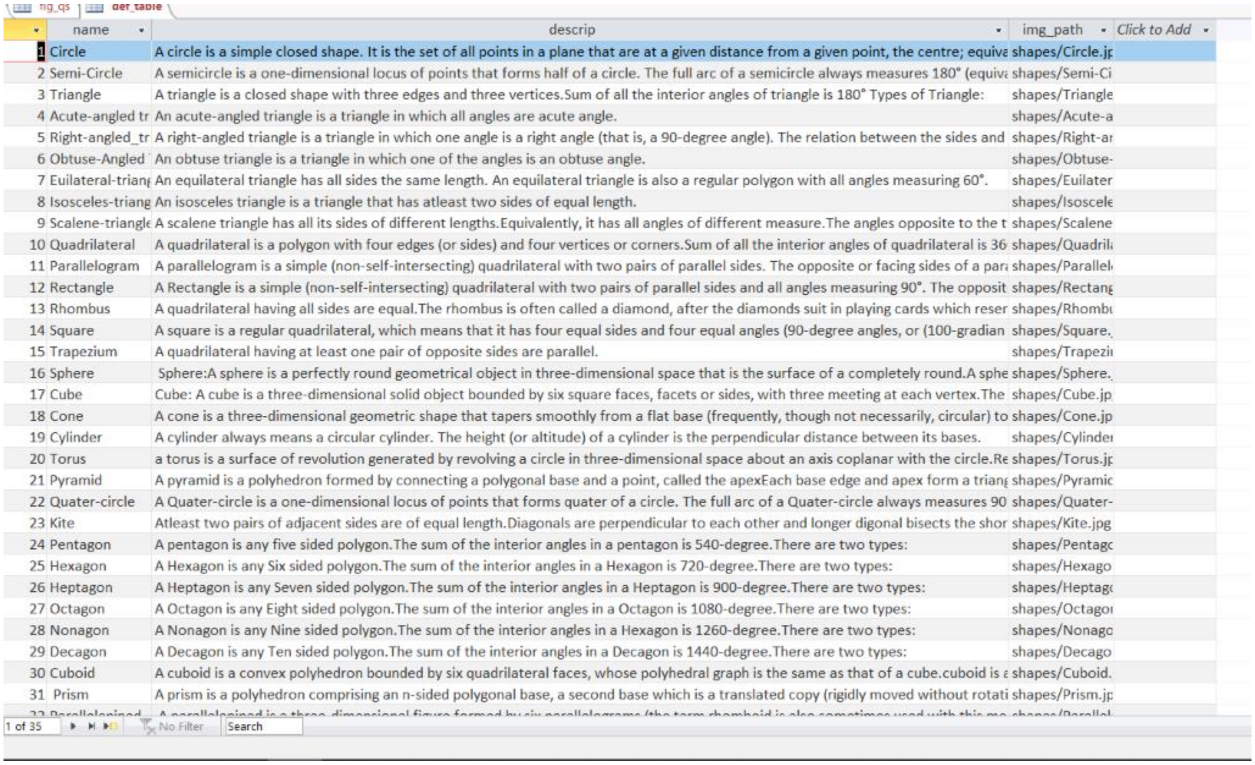
Question Bank sub-module : This has a bunch of question for testing his/her progress in understanding of the shapes which he/she needs to solve in stipulated time so as to score marks .

Game : This section is of immense interest with very adorable background and graphics where can enjoy and get a grip on the knowledge simultaneously .

Tree Diagram of the Module:

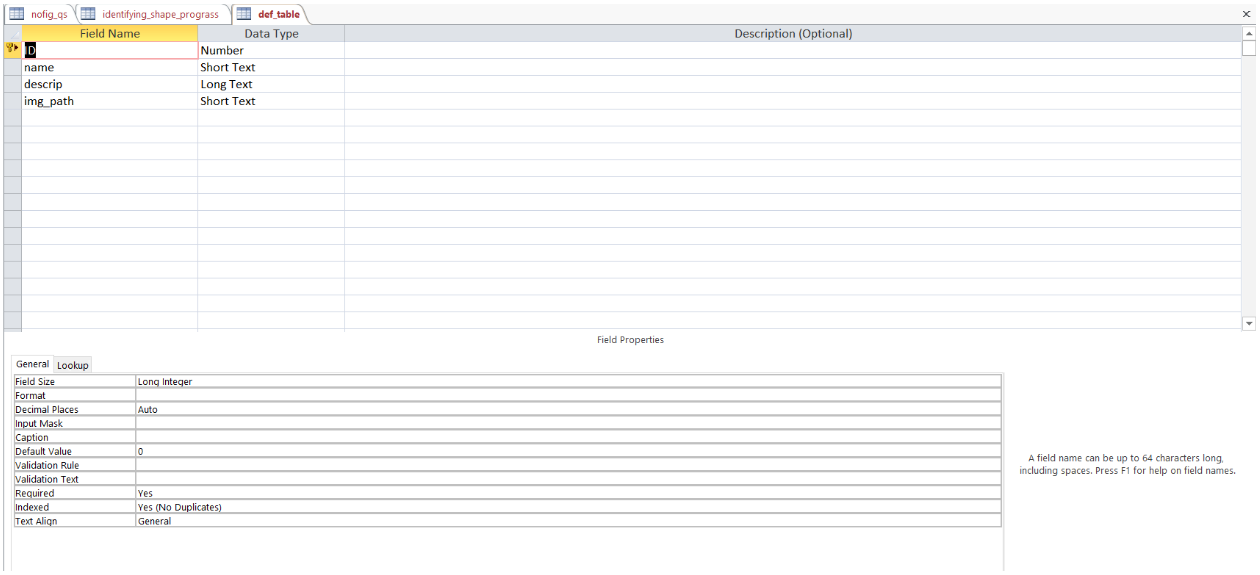


Database:



	name	descrip	img_path	Click to Add
1	Circle	A circle is a simple closed shape. It is the set of all points in a plane that are at a given distance from a given point, the centre; equiva	shapes/Circle.jp	
2	Semi-Circle	A semicircle is a one-dimensional locus of points that forms half of a circle. The full arc of a semicircle always measures 180° (equiva	shapes/Semi-Ci	
3	Triangle	A triangle is a closed shape with three edges and three vertices.Sum of all the interior angles of triangle is 180° Types of Triangle:	shapes/Triangle	
4	Acute-angled tr	An acute-angled triangle is a triangle in which all angles are acute angle.	shapes/Acute-a	
5	Right-angled_tr	A right-angled triangle is a triangle in which one angle is a right angle (that is, a 90-degree angle). The relation between the sides and	shapes/Right-ar	
6	Obtuse-Angled	An obtuse triangle is a triangle in which one of the angles is an obtuse angle.	shapes/Obtuse-	
7	Euilateral-triang	An equilateral triangle has all sides the same length. An equilateral triangle is also a regular polygon with all angles measuring 60°.	shapes/Euilater	
8	Isosceles-triang	An isosceles triangle is a triangle that has atleast two sides of equal length.	shapes/Isoscele	
9	Scalene-triangle	A scalene triangle has all its sides of different lengths.Equivalently, it has all angles of different measure.The angles opposite to the t	shapes/Scalene	
10	Quadrilateral	A quadrilateral is a polygon with four edges (or sides) and four vertices or corners.Sum of all the interior angles of quadrilateral is 36	shapes/Quadrila	
11	Parallelogram	A parallelogram is a simple (non-self-intersecting) quadrilateral with two pairs of parallel sides. The opposite or facing sides of a para	shapes/Parallel	
12	Rectangle	A Rectangle is a simple (non-self-intersecting) quadrilateral with two pairs of parallel sides and all angles measuring 90°. The opposit	shapes/Rectang	
13	Rhombus	A quadrilateral having all sides are equal.The rhombus is often called a diamond, after the diamonds suit in playing cards which reser	shapes/Rhombu	
14	Square	A square is a regular quadrilateral, which means that it has four equal sides and four equal angles (90-degree angles, or (100-gradian	shapes/Square._	
15	Trapezium	A quadrilateral having at least one pair of opposite sides are parallel.	shapes/Trapezi	
16	Sphere	Sphere:A sphere is a perfectly round geometrical object in three-dimensional space that is the surface of a completely round.A sphe	shapes/Sphere._	
17	Cube	Cube: A cube is a three-dimensional solid object bounded by six square faces, facets or sides, with three meeting at each vertex.The	shapes/Cube.jp	
18	Cone	A cone is a three-dimensional geometric shape that tapers smoothly from a flat base (frequently, though not necessarily, circular) to	shapes/Cone.jp	
19	Cylinder	A cylinder always means a circular cylinder. The height (or altitude) of a cylinder is the perpendicular distance between its bases.	shapes/Cylinder	
20	Torus	a torus is a surface of revolution generated by revolving a circle in three-dimensional space about an axis coplanar with the circle.Re	shapes/Torus.jp	
21	Pyramid	A pyramid is a polyhedron formed by connecting a polygonal base and a point, called the apexEach base edge and apex form a triang	shapes/Pyramic	
22	Quater-circle	A Quater-circle is a one-dimensional locus of points that forms quater of a circle. The full arc of a Quater-circle always measures 90	shapes/Quater-	
23	Kite	Atleast two pairs of adjacent sides are of equal length.Diagonals are perpendicular to each other and longer digonal bisects the shor	shapes/Kite.jpg	
24	Pentagon	A pentagon is any five sided polygon.The sum of the interior angles in a pentagon is 540-degree.There are two types:	shapes/Pentagc	
25	Hexagon	A Hexagon is any Six sided polygon.The sum of the interior angles in a Hexagon is 720-degree.There are two types:	shapes/Hexago	
26	Heptagon	A Heptagon is any Seven sided polygon.The sum of the interior angles in a Heptagon is 900-degree.There are two types:	shapes/Heptago	
27	Octagon	A Octagon is any Eight sided polygon.The sum of the interior angles in a Octagon is 1080-degree.There are two types:	shapes/Octagoı	
28	Nonagon	A Nonagon is any Nine sided polygon.The sum of the interior angles in a Hexagon is 1260-degree.There are two types:	shapes/Nonagc	
29	Decagon	A Decagon is any Ten sided polygon.The sum of the interior angles in a Decagon is 1440-degree.There are two types:	shapes/Decago	
30	Cuboid	A cuboid is a convex polyhedron bounded by six quadrilateral faces, whose polyhedral graph is the same as that of a cube.cuboid is a	shapes/Cuboid.	
31	Prism	A prism is a polyhedron comprising an n-sided polygonal base, a second base which is a translated copy (rigidly moved without rotati	shapes/Prism.jp	
32	Parallelopiped	A parallelopiped is a three-dimensional figure formed by six parallelograms (the term rhomboid is also sometimes used with this me	shapes/Parallel	

1 of 35 No Filter Search

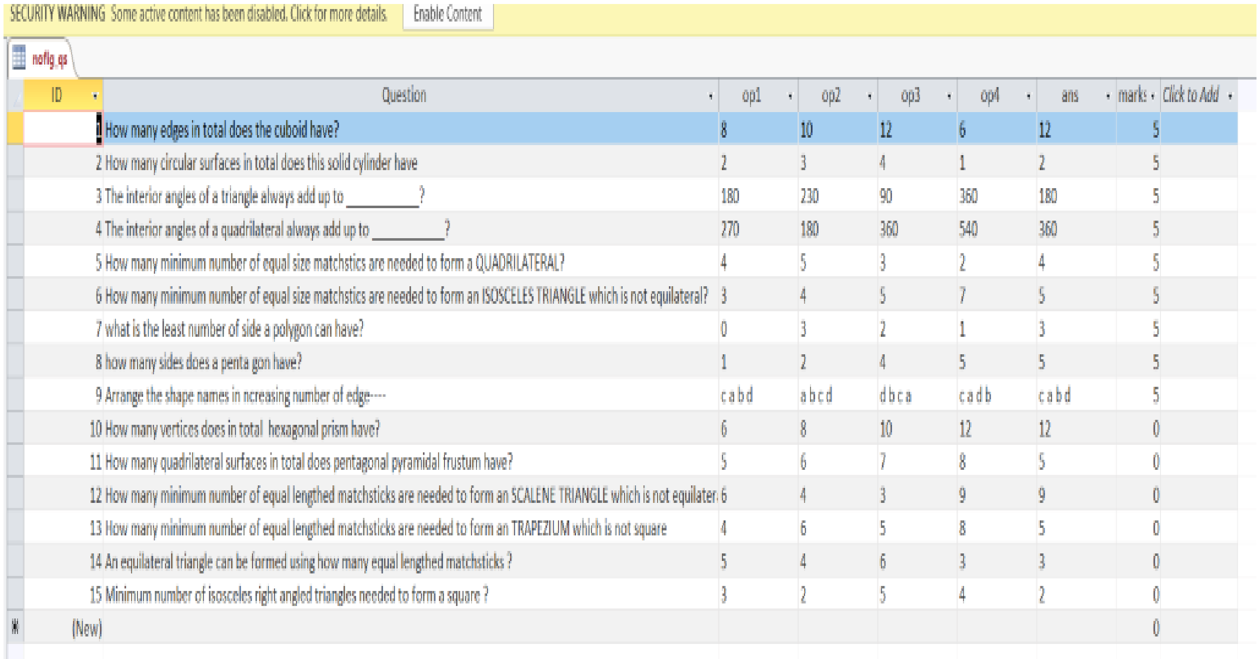


Field Name	Data Type	Description (Optional)
ID	Number	
name	Short Text	
descrip	Long Text	
img_path	Short Text	

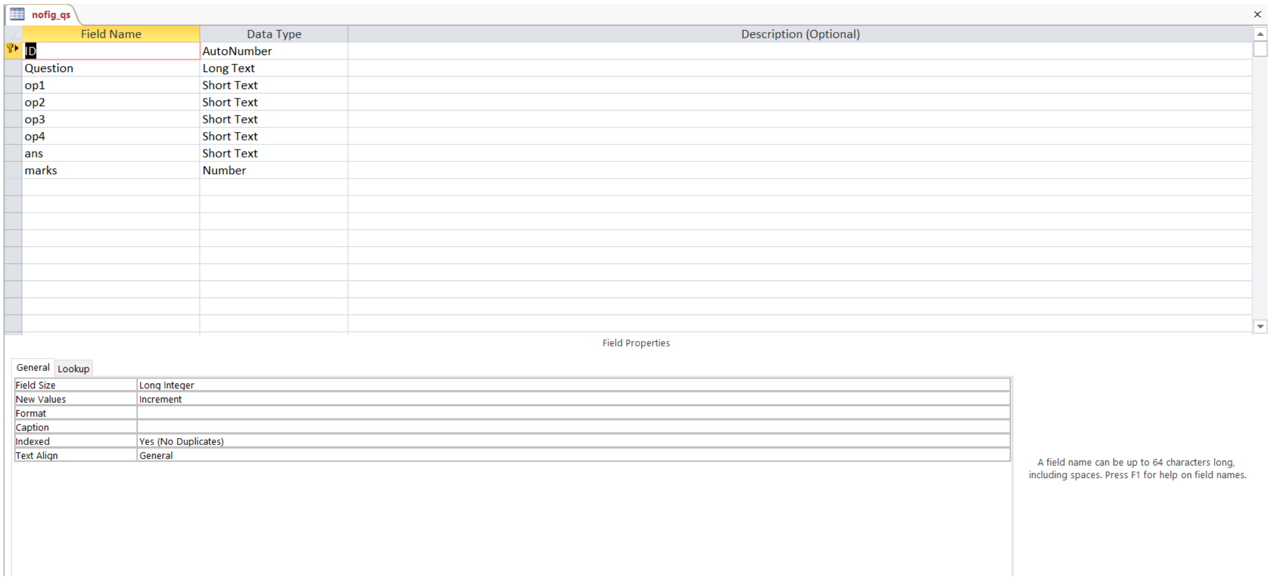
Field Size	Long Integer
Format	
Decimal Places	Auto
Input Mask	
Caption	
Default Value	0
Validation Rule	
Validation Text	
Required	Yes
Indexed	Yes (No Duplicates)
Text Align	General

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

a)Database containing shape definitions and information along with there image is shown below.


SECURITY WARNING Some active content has been disabled. Click for more details.
Enable Content
nofig_qs

ID	Question	op1	op2	op3	op4	ans	marks	Click to Add
1	How many edges in total does the cuboid have?	8	10	12	6	12	5	
2	How many circular surfaces in total does this solid cylinder have	2	3	4	1	2	5	
3	The interior angles of a triangle always add up to _________?	180	230	90	360	180	5	
4	The interior angles of a quadrilateral always add up to _________?	270	180	360	540	360	5	
5	How many minimum number of equal size matchstics are needed to form a QUADRILATERAL?	4	5	3	2	4	5	
6	How many minimum number of equal size matchstics are needed to form an ISOSCELES TRIANGLE which is not equilateral?	3	4	5	7	5	5	
7	what is the least number of side a polygon can have?	0	3	2	1	3	5	
8	how many sides does a penta gon have?	1	2	4	5	5	5	
9	Arrange the shape names in ncreasing number of edge----	c a b d	a b c d	d b c a	c a d b	c a b d	5	
10	How many vertices does in total hexagonal prism have?	6	8	10	12	12	0	
11	How many quadrilateral surfaces in total does pentagonal pyramidal frustum have?	5	6	7	8	5	0	
12	How many minimum number of equal lengthed matchsticks are needed to form an SCALENE TRIANGLE which is not equilater	6	4	3	9	9	0	
13	How many minimum number of equal lengthed matchsticks are needed to form an TRAPEZIUM which is not square	4	6	5	8	5	0	
14	An equilateral triangle can be formed using how many equal lengthed matchsticks ?	5	4	6	3	3	0	
15	Minimum number of isosceles right angled triangles needed to form a square ?	3	2	5	4	2	0	
(New)							0	



nofig_qs

Field Name	Data Type	Description (Optional)
ID	AutoNumber	
Question	Long Text	
op1	Short Text	
op2	Short Text	
op3	Short Text	
op4	Short Text	
ans	Short Text	
marks	Number	

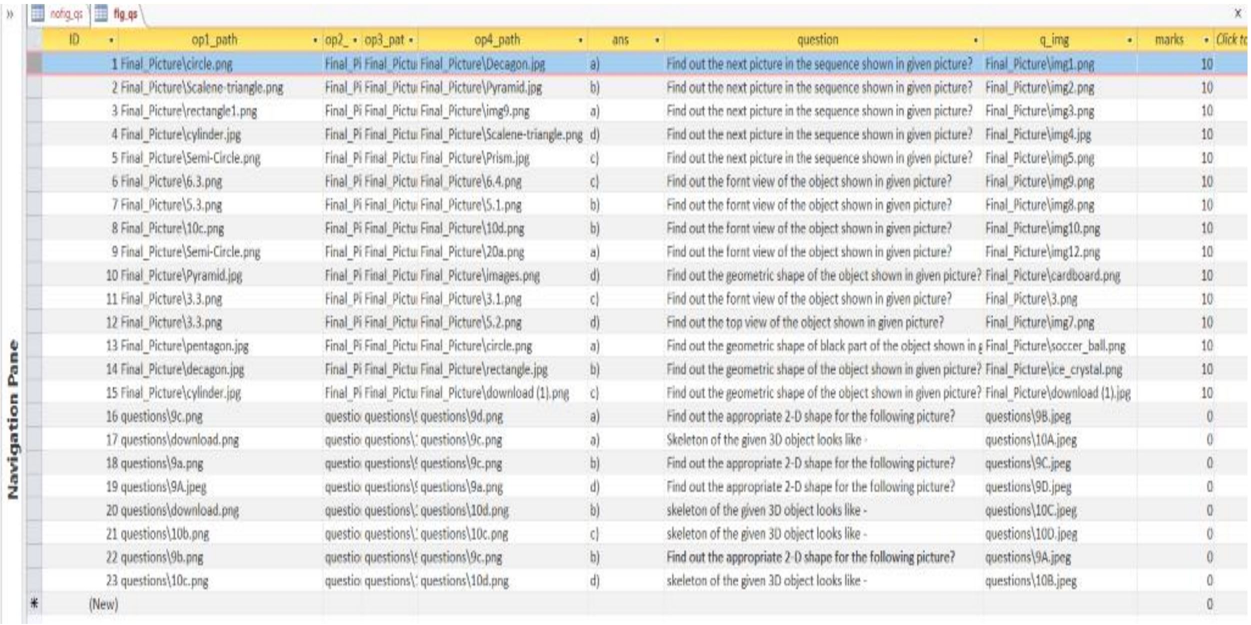
Field Properties

General Lookup

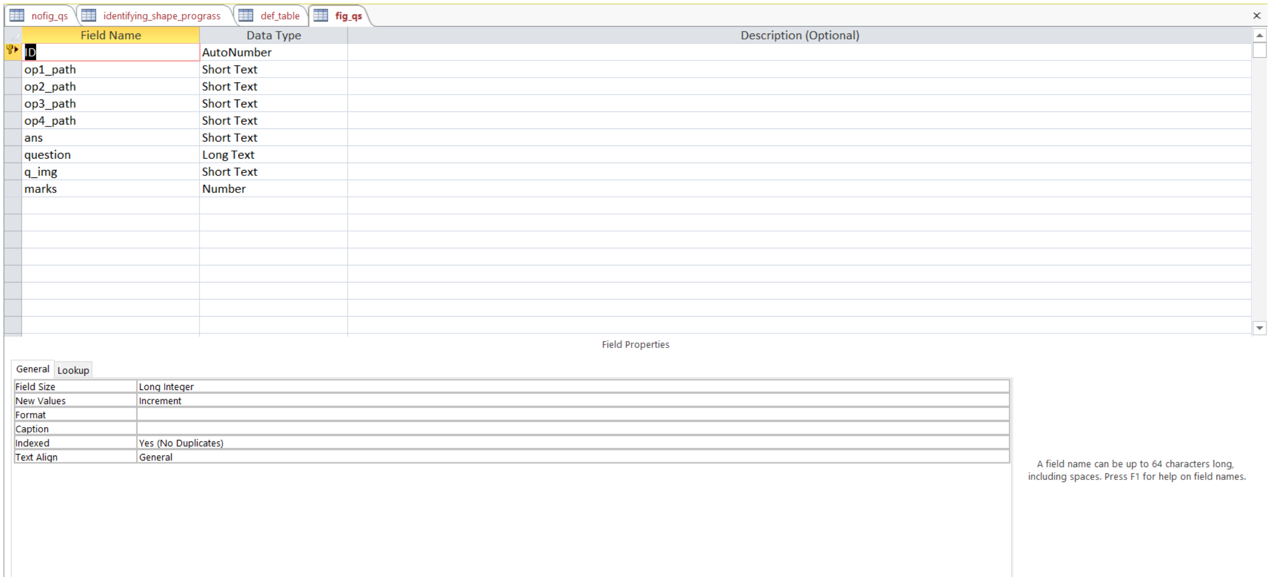
Field Size	Long Integer
New Values	Increment
Format	
Caption	
Indexed	Yes (No Duplicates)
Text Align	General

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

b) Database for question with no figures .



ID	op1_path	op2_	op3_pat	op4_path	ans	question	q_img	marks	Click to
1	Final_Picture\circle.png	Final_Pi	Final_Pictu	Final_Picture\Decagon.jpg	a)	Find out the next picture in the sequence shown in given picture?	Final_Picture\img1.png	10	
2	Final_Picture\Scalene-triangle.png	Final_Pi	Final_Pictu	Final_Picture\Pyramid.jpg	b)	Find out the next picture in the sequence shown in given picture?	Final_Picture\img2.png	10	
3	Final_Picture\rectangle1.png	Final_Pi	Final_Pictu	Final_Picture\img9.png	a)	Find out the next picture in the sequence shown in given picture?	Final_Picture\img3.png	10	
4	Final_Picture\cylinder.jpg	Final_Pi	Final_Pictu	Final_Picture\Scalene-triangle.png	d)	Find out the next picture in the sequence shown in given picture?	Final_Picture\img4.jpg	10	
5	Final_Picture\Semi-Circle.png	Final_Pi	Final_Pictu	Final_Picture\Prism.jpg	c)	Find out the next picture in the sequence shown in given picture?	Final_Picture\img5.png	10	
6	Final_Picture\6.3.png	Final_Pi	Final_Pictu	Final_Picture\6.4.png	c)	Find out the fornt view of the object shown in given picture?	Final_Picture\img9.png	10	
7	Final_Picture\5.3.png	Final_Pi	Final_Pictu	Final_Picture\5.1.png	b)	Find out the fornt view of the object shown in given picture?	Final_Picture\img8.png	10	
8	Final_Picture\10c.png	Final_Pi	Final_Pictu	Final_Picture\10d.png	b)	Find out the fornt view of the object shown in given picture?	Final_Picture\img10.png	10	
9	Final_Picture\Semi-Circle.png	Final_Pi	Final_Pictu	Final_Picture\20a.png	a)	Find out the fornt view of the object shown in given picture?	Final_Picture\img12.png	10	
10	Final_Picture\Pyramid.jpg	Final_Pi	Final_Pictu	Final_Picture\images.png	d)	Find out the geometric shape of the object shown in given picture?	Final_Picture\cardboard.png	10	
11	Final_Picture\3.3.png	Final_Pi	Final_Pictu	Final_Picture\3.1.png	c)	Find out the fornt view of the object shown in given picture?	Final_Picture\3.png	10	
12	Final_Picture\3.3.png	Final_Pi	Final_Pictu	Final_Picture\5.2.png	d)	Find out the top view of the object shown in given picture?	Final_Picture\img7.png	10	
13	Final_Picture\pentagon.jpg	Final_Pi	Final_Pictu	Final_Picture\circle.png	a)	Find out the geometric shape of black part of the object shown in g	Final_Picture\soccer_ball.png	10	
14	Final_Picture\decagon.jpg	Final_Pi	Final_Pictu	Final_Picture\rectangle.jpg	b)	Find out the geometric shape of the object shown in given picture?	Final_Picture\ice_crystal.png	10	
15	Final_Picture\cylinder.jpg	Final_Pi	Final_Pictu	Final_Picture\download (1).png	c)	Find out the geometric shape of the object shown in given picture?	Final_Picture\download (1).jpg	10	
16	questions\9c.png	questio	questions\	questions\9d.png	a)	Find out the appropriate 2-D shape for the following picture?	questions\9B.jpeg	0	
17	questions\download.png	questio	questions\	questions\9c.png	a)	Skeleton of the given 3D object looks like -	questions\10A.jpeg	0	
18	questions\9a.png	questio	questions\	questions\9c.png	b)	Find out the appropriate 2-D shape for the following picture?	questions\9C.jpeg	0	
19	questions\9A.jpeg	questio	questions\	questions\9a.png	d)	Find out the appropriate 2-D shape for the following picture?	questions\9D.jpeg	0	
20	questions\download.png	questio	questions\	questions\10d.png	b)	skeleton of the given 3D object looks like -	questions\10C.jpeg	0	
21	questions\10b.png	questio	questions\	questions\10c.png	c)	skeleton of the given 3D object looks like -	questions\10D.jpeg	0	
22	questions\9b.png	questio	questions\	questions\9c.png	b)	Find out the appropriate 2-D shape for the following picture?	questions\9A.jpeg	0	
23	questions\10c.png	questio	questions\	questions\10d.png	d)	skeleton of the given 3D object looks like -	questions\10B.jpeg	0	
(New)								0	



Field Name	Data Type	Description (Optional)
ID	AutoNumber	
op1_path	Short Text	
op2_path	Short Text	
op3_path	Short Text	
op4_path	Short Text	
ans	Short Text	
question	Long Text	
q_img	Short Text	
marks	Number	

Field Properties

General Lookup

Field Size	Long Integer
New Values	Increment
Format	
Caption	
Indexed	Yes (No Duplicates)
Text Align	General

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

c) Database for the questions with figure .

Functions

Game (game.h) :

Private: System::Void game_Load(System::Object^ sender, System::EventArgs^ e): Funtion call to start the game from the beginning.

Private: System::Void game_Paint(System::Object^ sender, System::Windows::Forms ::PaintEventArgs^ e): Function to paint the background of the form.

g->DrawImage(Image::FromFile("game_pic/pic9.png"), p - 1200, 300): This enable the designer to draw the image in the windows form as there in the mentioned file

g->FillPie(Brushes::Red, p- 140, 460, 120, 120, 45, 300): This function fills the pie with red colour .

Point(Convert::ToInt16(label5->Text) + slide + 10, Convert::ToInt16(label6->Text)): Function to record the clicked positon of the mouse to test whether it lies in the shape of interest .

Textbox->Text = Reads/Writes from/to the Textbox named TextBox.

Label->Text = Reads/Writes from/to the Label named Lable.

g->DrawPolygon(Pens::Red, p9): Function to draw a polygoan with number of points in the array p9 .

Private: System::Void timer1_Tick(System::Object^ sender, System::EventArgs^ e): Function to change the background picture i.e. graphics of the windows form .

Private: System::Double cirpt(Point p1, Point p, double r): Function to find point where user clicked (i.e. point p) lies in the circle with center ‘p1’ and radius ‘r’ or not

Private:System::Double area(Point p1, Point p2, Point p3): Function to findout the area formed by the triangle having the points p1,p2,p3 .

Private:System::Double tript(Point p1, Point p2, Point p3, Point p): Function to find point where user clicked (i.e. point p) lies in the triangle with points p1,p2,p3 or not

Private:System::Double recpt(Point p1, Point p2, Point p3, Point p4, Point p): Function to find point where user clicked (i.e. point p) lies in the rectangle with points p1,p2,p3,p4 or not

timer1->Stop(): Function to stop the timer1 .

panel1->Show(): Function to display the panel1 .

Private: System::Void Exit_Click(System::Object^ sender, System::EventArgs^ e) : Calling the exit function to exit from the game.

Private: System::Void pause_play_Click(System::Object^ sender, System::EventArgs^ e): Function to control the pause and play button of the game .

Private: System::Void timer3_Tick(System::Object^ sender, System::EventArgs^ e): Function to display the score after the game is over .

Private: System::Void initiate() : Function to start the game again and reset the things after the defeat .

Private: System::Void ok_Click(System::Object^ sender, System::EventArgs^ e): Function to start the game after the user reads the instruction and click on ‘ok’ button .

View All Shapes (all_shape.h):

#include"shape_defination.h" : This includes the header file “shape_defination.h” which ha sthe information of the shapes .

this->panel1->SuspendLayout(): The layout logic of the control is suspended until the ResumeLayout method is called.

System::Drawing::Point(385, 271) : Represents an ordered pair of integer x- and y-coordinates that defines a point in a two-dimensional plane private: System::Void picsload(): In this function pictures are added to picturebox from a file shapes using imagelocation

pictureBox25->ImageLocation = "shapes/Pentagon.jpg": Add the image from mentioned location to the mentioned picture box .

private: System::Void all_shape_Load(System::Object^ sender, System::EventArgs^ e): Adjust the position of the panels .

private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) : Button click event to go to previous page of the shape information by sliding the panel from left to right.

private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e): Button click event to go to next page of the shape information by sliding the panel from right to left .

Textbox->Text = Reads/Writes from/to the Textbox named TextBox.

Label->Text = Reads/Writes from/to the Label named Lable.

private: System::Void timer1_Tick(System::Object^ sender, System::EventArgs^ e) , private: System::Void timer2_Tick(System::Object^ sender, System::EventArgs^ e): Controls the movement of the panel .

private: System::Void go_to_learn() : Function to call definition form (information of the shape form) .

private: System::Void pictureBox1_Click(System::Object^ sender, System::EventArgs^ e): Event that redirect the user on clicking specific picture box to its corresponding information page.

this->Close() = Will close the 'view all shapes' form .

Learn Shapes (shape_definition.h):

this->richTextBox2->BackColor = System::Drawing::Color::Snow : Sets the background colour of the richTextBox2 to snow .

this->richTextBox2->Location = System::Drawing::Point(42, 227): Fixes the location of the richTextBox2 .

this->AutoScaleMode=System::Windows::Forms::AutoScaleMode:: Font:
An AutoScaleMode that represents the current scaling mode for fonts.

Timer->Start() : Starts the timer.

panel->Hide() : Hides out the panel named as panel.

Label->Text = Reads/Writes from/to the Label named Lable.

private: System::Void shape_defination_Load(System::Object^ sender, System::EventArgs^ e): Adjust the positioning of the panels .

private: System::Void button4_Click(System::Object^ sender, System::EventArgs^ e): Function to go to the corresponding shape information whoes number is entered in the textbox by the user.

private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) : Button click event to go to previous page of the shape information by sliding the panel from left to right.

private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e): Button click event to go to next page of the shape information by sliding the panel from right to left .

private: System::Void timer1_Tick(System::Object^ sender, System::EventArgs^ e) , private: System::Void timer2_Tick(System::Object^ sender, System::EventArgs^ e): Controls the movement of the panel .

private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e): This function exits the 'learn shapes' form .

Draw Shapes (drawshap.h):

private: System::Void drawshap_Load(System::Object^ sender, System::EventArgs^ e): Function to initialize the 'Draw Shape' form by declaring array of points for storing the joined points . The pen color and width for drawing the lines .

private: System::Void drawshap_Paint(System::Object^ sender, System::Windows::Forms:: PaintEventArgs^ e): Drawing small circles as points for drawing different shapes .

private: System::Void drawshap_MouseClick(System::Object^ sender, System::Windows:: Forms::MouseEventArgs^ e) : sending mouse clicked x and y coordinates to label2 and label3.

Label->Text = Reads/Writes from/to the Label named Lable.

private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e): To refresh the drawing board .

private: System::Void check_lastone() : Function to check if same point is clicked more than once if so keep only one in local array and remove other .

private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e): This function checks whether first and the last point clicked are same or not and if they are not same it returns "not defined" . And for Three or more consecutive collinear points it removes intermediate points and only stores the endpoints . This checks for the interesting lines if any leading to "not defined" state. At last it decides the type of polygoaqn and displays its name in the textbox .

private: System::Double get_dist(Point p1, Point p2) : Finds the distance between two points p1 and p2 .

private: System::Void check_last3() : Back tracing of the drawn path is restricted .

private:System::Double cirpt(Point p1, Point p, double r) : This checks whether user properly clicked the point while joining them by lines .

private: System::Void im_pt(Point A, Point B) : Function to insert intermediate points between A to B .

private: System::Boolean colin(Point A, Point B, Point C) : Boolean function to check are the points A, B, C collinear .

private:System::Double orientation(Point A, Point B, Point C): Checks the orientation of the points A,B,C (i.e. CLW or ACLW)

private:System::Boolean dointersect(Point p1, Point q1, Point p2, Point q2): Checks whether seg p1q1 and p2q2 intersect .

private:System::Int32 max(int a,int b) , private:System::Int32 min(int a, int b) : Determines the minimum and maximum of a and b .

private:System::Boolean onSegment(Point p, Point q, Point r) : Checks if point r lies on seg pq.

private:System::Boolean check_intersection() : Checks the intersection of the sides with every other side of the drawn shape by the user .

private:System::Boolean isparallel(Point A, Point B, Point C, Point D) : Checking AB || CD or not .

private:System::Boolean isperp(Point A, Point B, Point C) : Checking AB ,BC perpendicular or not .

private:System::Void tri_type() : This function decides the type of triangle .

private:System::Void rec_type(): Decides the type of quadrilateral .

Question Bank :

Question_no_Figure (question_no_figure.h) :

this->richTextBox1->Size = System::Drawing::Size(965, 187) Setting the size of the richtextbox1(used to display options of the question) .

this->panel3->Size = System::Drawing::Size(1002, 243) : Setting the size of the panel1 .

private: System::Void checkBox1_CheckStateChanged(System::Object^ sender, System::EventArgs^ e) : Checkbox for option 1.

private: System::Void check_Click(System::Object^ sender, System::EventArgs^ e) : Function to check the ticked answer by the user .(This function go for checking after user click on check button)

private: System::Void timer3_Tick(System::Object^ sender, System::EventArgs^ e) : Function to display the time left for solving the question .

private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e) : Function to got to next question by clicking next button .

private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) : function to exit from this form .

private: System::Void save() : Function for saving all records before exit .

private:System::Void go() : Function to load question option from database showing them in the textbox and richtextbox described above .

private: System::Void question_no_figure_Load(System::Object^ sender, System::EventArgs^ e) : Function to load this form by initializing the position of panel , richtextbox , textboxes and checkboxes .

Question_with_Figure (question_with_figure.h) :

PictureBox1: Used to display figure in the question .

PictureBox2, PictureBox3, PictureBox4, PictureBox5 : Used to show 4 options for the question .

TextBox1 : Will show 'correct' if the clicked answer is correct otherwise will show 'wrong' .

Label->Text = Reads/Writes from/to the Label named Lable.

private: System::Void pictureBox2_Click(System::Object^ sender, System::EventArgs^ e) Function to record and examine the response of the user when he/she click PictureBox1 .

private: System::Void timer2_Tick(System::Object^ sender, System::EventArgs^ e) : Function to display the time left for solving the question . If not solved in given time will go to next question automatically .

private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e) : Function for button click 'next' which take the user to the next question before the given time .

private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) : function to exit from this form .

private: System::Void save() : Function for saving all records before exit .

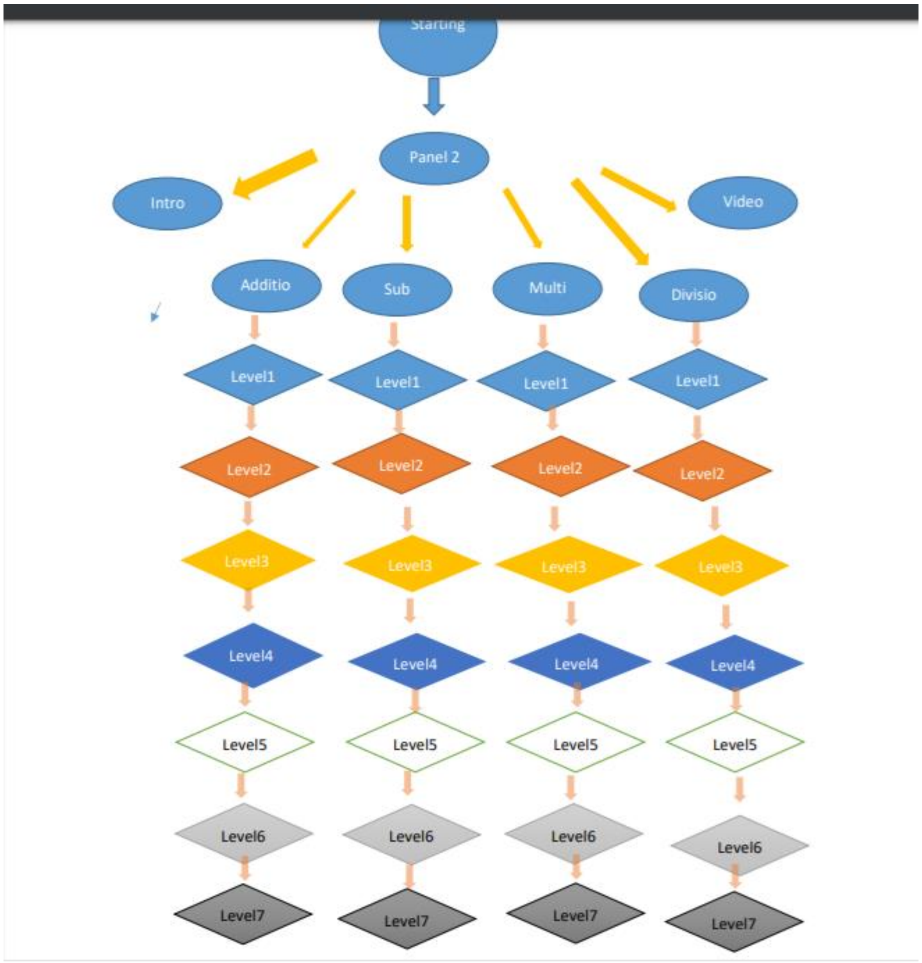
private:System::Void go() : Function to load question option from database showing them in the textbox and richtextbox described above .

private: System::Void question_with_figure_Load(System::Object^ sender, System::EventArgs^ e) : Function to load this form by initializing the position of panel , richtextbox , textboxes and checkboxes .

Abacus:

The movements of beads

FlowChart



Movement of the bead is controlled by the following algorithm

```
{
  IF( the location of the bead is down then change the position to up and change the whole beads locations )
  ELSE IF (the location of bead is up then change the bead location to down and change the location of other beads accordingly).
}
```

The beads are hardcoded by their position in (x.y) plane.

The given software has 4 panels

- Panel 1 : Contains introductory Form1
- Panel 2 : Contains the abacus used for softcopy

- Panel 3: Contains the questions used to answer

Generation of Questions

The questions are generated using a rand() function, so their may be repeatition in the questions.

The numbers are generated of the form   b=rand()%5+1.

As the levels go on the complexity of the questions become higher and higher.

For the first 3 Levels the numbers generated are upto 2 digits and as the level is increased to Level 4,5,6,7 the no of digits in problems increases from 2,2,3,3 resp.

The timer is set at 2 min for reset every time you click “Submit” button.

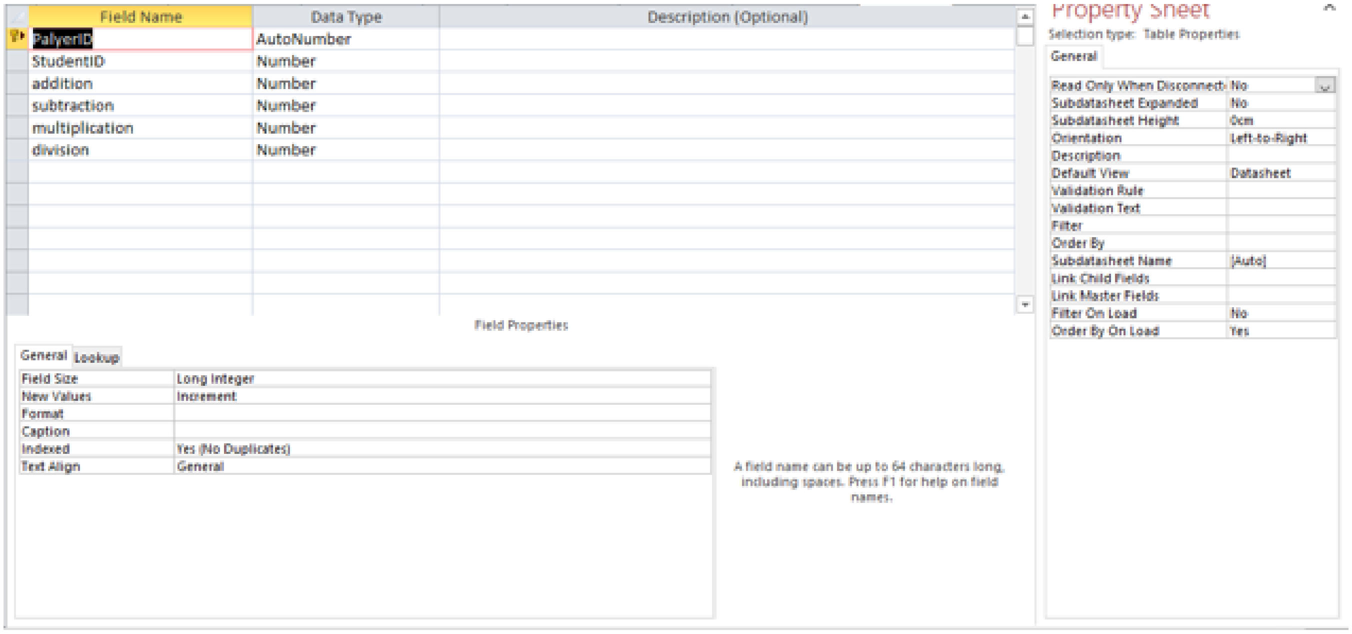
The abacus to solve the Question

In the questions we have a 4 Bit abacus so the place value is as shown of the beads,

1000th 100th 10th 1th   Place digit in abacus

The order of MSB to LSB is from left to Right.

Database:



Field Name	Data Type	Description (Optional)
PalyerID	AutoNumber	
StudentID	Number	
addition	Number	
subtraction	Number	
multiplication	Number	
division	Number	

Field Properties

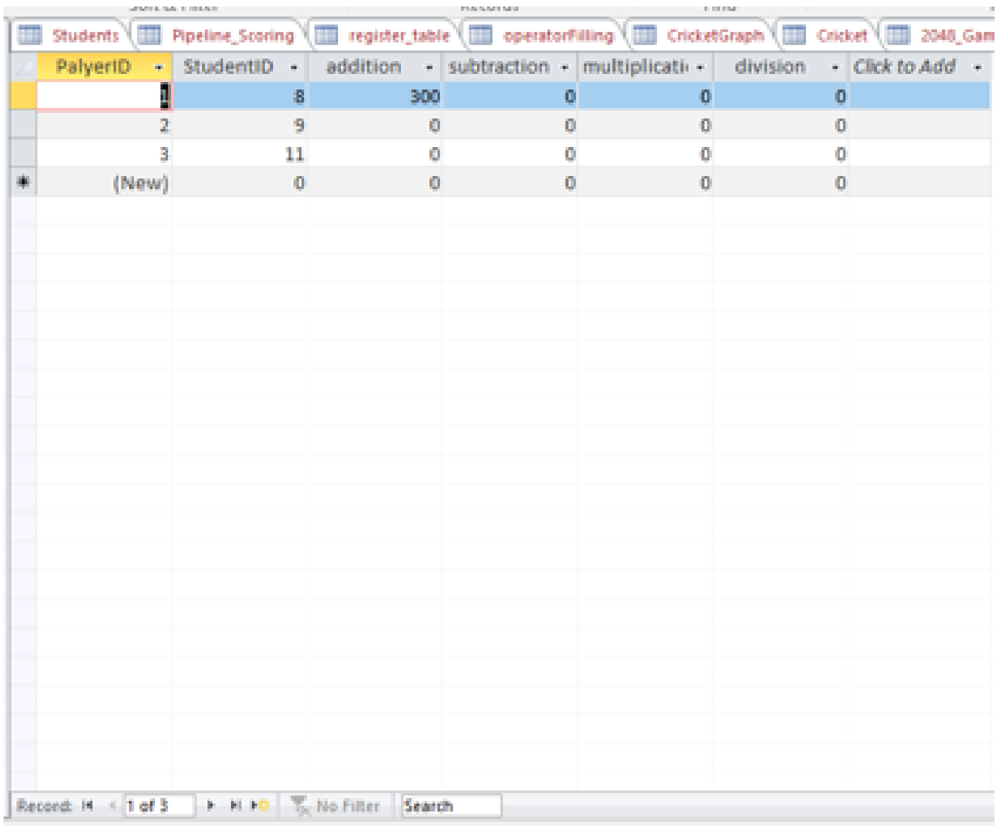
Field Size	Long Integer
New Values	Increment
Format	
Caption	
Indexed	Yes (No Duplicates)
Text Align	General

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

Property Sheet

Selection type: Table Properties

Read Only When Disconnect	No
Subdatasheet Expanded	No
Subdatasheet Height	0cm
Orientation	Left-to-Right
Description	
Default View	Datasheet
Validation Rule	
Validation Text	
Filter	
Order By	
Subdatasheet Name	[Auto]
Link Child Fields	
Link Master Fields	
Filter On Load	No
Order By On Load	Yes



PalyerID	StudentID	addition	subtraction	multiplicati	division	Click to Add
1	8	300	0	0	0	
2	9	0	0	0	0	
3	11	0	0	0	0	
(New)	0	0	0	0	0	