CMPE 230 – SYSTEMS PROGRAMMING
Project 3

Author: **Ali BATIR**
Student ID: **2015400261**

Author: **Kaan ŞENPARLAK**
Student ID: **2016401189**

Introduction

In this project a Find the Pair game was implemented with Qt for C++.Find the pair games are games that require the player to match similar elements. As the name implies, the player needs to find a match for the card. For example, in this project 12 letters (24 cards in total) are placed face down in random order. The player turns over two cards at a time, with the goal of turning over a matching pair, by using their memory.

Pairs	5	Tries	11	Reset	
X	X	X	X		X
		X		A	X
X		A			X
X	X		X	X	X

Implementation

```
QApplication app(argc, argv); //QApplication
QMainWindow *window = new QMainWindow(); // QMainWindow


    window->setWindowTitle(QString::fromUtf8("Find The Pair")); // setting window's title as Card Match Game
    window->resize(600, 300); // resize window for table

    QWidget *centralWidget = new QWidget(window); //QWidget
```

UI is constructed without the QT creator and designed without it afterwards.

```
//vector for storing letters as QString
vector<QString> letterVec = {"A","A","B","B","C","C","D","D","E","E","F","F",
                             "G","G","H","H","I","I","J","J","K","K","L","L"};

//stack for letters as QString to push randomly from vector of letter
stack<QString> letterStack ;

MyButtonGroup* group = new MyButtonGroup(centralWidget);    //MyButtonGroup class initialization

int size  = letterVec.size();           //size of vector of letter     ⚠implicit conversion loses integer precision: 'std
for(int i= 0 ; i<size ; i++){           // for loop to push letter QString stack from vector
    int index = rand()%size;                    //get random index according to vector size
    letterStack.push(letterVec[index]);         //get letter with random index of vector of QString and push it to stack
    letterVec.erase(letterVec.begin() + index);     //erase the letter of that index in vector
    i=-1 ;  // decrement i to fit in vector index
    size-- ; // decrement vector size
}
```

A letter vector and a letter stack is constructed and the letters are shuffled into the stack.

```
QTableWidget *table = new QTableWidget(); //QTableWidget
MyButtonGroup::mytable = table ;      //Assign table pointer to mytable from MyButtonGroup
QTableWidgetItem *tableItem ;       // QTableWidgetItem
table->verticalHeader()->setVisible(false);     // set invisible to vertical header of table to remove numbers
table->horizontalHeader()->setVisible(false);   // set invisible to horizontal header of table to remove numbers

    table->setRowCount(5);     // set row number to 5 according given table in description
    table->setColumnCount(6); // set column number

    table->setSizePolicy(QSizePolicy::Expanding,QSizePolicy::Expanding); //set size policy for cells in table
           QPushButton *reset = new QPushButton("Reset"); //QPushButton pointer for reset
           MyButtonGroup::reset = reset; //Assign reset pointer to reset QPushButton from MyButtonGroup

           QLabel *triesNum =new QLabel("0");
           MyButtonGroup::triesNum = triesNum;
           QLabel *pairsNum =new QLabel("0");
           MyButtonGroup::pairsNum = pairsNum;

           QLabel *pairs =new QLabel("Pairs");
           QLabel *tries =new QLabel("Tries");
           QLabel *resetLabel =new QLabel("Reset");

           table->setRowCount(5);
           table->setColumnCount(6);
           table->setCellWidget(0,0,pairs);
           table->setCellWidget(0,1,pairsNum);
           table->setCellWidget(0,2,tries);
           table->setCellWidget(0,3,triesNum);
           table->setCellWidget(0,4,resetLabel);

           table->cellWidget(0,0)->resize(50,50);
           table->setSpan(0,4,1,2);

           table->setSizePolicy(QSizePolicy::Expanding,QSizePolicy::Expanding);
           table->setIndexWidget(table->model()->index(0, 4), reset);
           table->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);
           table->verticalHeader()->setSectionResizeMode(QHeaderView::Stretch);
```

The main widget of the design, the button table is designed and the necessary adjustments are made. Buttons are coded such that they enlarge and shrink whenever the window size changes.

```
    //add QPushButtons to table with nested for loop
for(int i=0 ; i<4 ; i++){
    for(int j=0 ; j<6 ; j++){
        QPushButton * buttonX = new QPushButton("X",centralWidget); //QPushButton pointers with text "X"
        tableItem= new QTableWidgetItem();//initalize new QTableWidgetItem in every loop

        MyButtonGroup::st[i][j] = letterStack.top(); // add the letters from randomly pushed stack to array from MyButtonGroup class
        letterStack.pop(); // pop letter that assigned to array above from stack

        MyButtonGroup::arr[i][j] = buttonX ; // add buttons to array from MyButtonGroup


        table->setItem(i+1,j,tableItem); //set table item to table
        table->setCellWidget(i+1, j, buttonX); // add buttons to table
        group->addButton(buttonX); // add buttons to MyButtonGroup class to identify which button is clicked

        if(letterStack.empty()) // if stack is empty breaks the loop
            break;

    }
}
```

The buttons are created and they are assigned to the stack of the letters which was shuffled previously.

```
if(but == reset){    // looks if clicked button is reset button
    vector<QString> letterShuffle ; //vector of QString to used when reset button is clicked
    //assign -1 to previous ids while starting new game
    prevId1 = -1 ;
    prevId2 = -1 ;

    //assign false to clicked flag
    clicked = false ;
    playerId = 1 ; // assign 1 to player id to start new game from player1

    //nested for loop to assign all letter cards(buttons) as "X" while starting new game
    for(int i=0 ; i<4 ; i++){
        for(int j=0 ; j<6 ; j++){
            arr[i][j]->setText("X"); //setting text for each button as "X"
            arr[i][j]->setEnabled(true); //set enable true for making button clickable
            letterShuffle.push_back(st[i][j]); // shuffle the array to assign letters to buttons randomly
        }
    }
    random_shuffle(letterShuffle.begin(),letterShuffle.end());  // shuffle the vector to assign letters to buttons randomly

    //nested for loop to assign new shuffled letter vector to letter array(st)
    for(int i=0 ; i<4 ; i++){
        for(int j=0 ; j<6 ; j++){
            st[i][j] = letterShuffle.back();
            letterShuffle.erase(letterShuffle.end()-1);
        }
    }
      countTry=0;
      countPair=0;
      triesNum->setText("0");
      pairsNum->setText("0");
    return ;
}
```

In this part, the reset button is implemented. When the reset button is pressed, all the letters are gathered again in the stack and reshuffled. Try and pair counts are reset and all the cards are again face down ("X").

```
        //compares the letters with first one that we stored  , if they match increment the player's score
    if(prev.compare(st[i][j]) == 0 ){
        but->setText(""); //set text empty if letters are matched
        but->setEnabled(false);  // set enabled false to prevent button clicked
        arr[prevId1][prevId2]->setEnabled(false); // set enabled false to prevent button clicked
         arr[prevId1][prevId2]->setText(""); //set text empty if letters are matched

         //count number of pair
          countPair++;
          QString s = QString::number(countPair); //convert int to string
          pairsNum->setText(s);
          //count number of try
           countTry++;
           QString s2 = QString::number(countTry); //convert int to string
           triesNum->setText(s2);
    }
    else{
        but->setText(st[i][j]); // set text to button from letter array
        QApplication::processEvents(); //process events

        usleep(500000); //using time sleep to see the two cards open a little time
        but->setText("X"); // set text "X" to that button if cards do not match
        if(prevId1 != -1 && prevId2 != -1)
             arr[prevId1][prevId2]->setText("X"); // set text "X" to other button if cards dont match
        prev = " "; // assign the empty prev string for new round

        //count number of try
         countTry++;
         QString s3 = QString::number(countTry); //convert int to string
         triesNum->setText(s3);
    }
    //assign -1 to previous ids  for new round
     prevId1 = -1 ;
     prevId2 = -1 ;
```

This part takes care of the core of the game, namely matchmaking. The first if statement is activated when there is a match. It makes the buttons disabled and changes their text. It also increases the pair count. The else statement deals with the no-match situation and performs necessary actions. Pressed button numbers are set to -1 at the end of the loop in order to prevent a conflict.