

[CROSSWORD]

إشراف المهندس: فراس سلمان

إعداد الطلاب

زكريا الشيخ – عبد الرحمن الحوراني – لبنى ديوب -
حنين مسعود – محمد عيد التل

الكلمات المتقاطعة

Crossword puzzle

abstract :

- crossword puzzle are today's most popular word game , however , their production by hand is a difficult and time consuming process , and the automation of this process has interested computer scientist from time to time , in this report we will mention a bit result of the search in these day , and then the target and importance of crossword puzzle , and how can benefit other , and then introduce a crossword puzzle algorithm which we use and other helper _process , and we will talk about our work and other , and we can improve then.....

- لعبة الكلمات المتقاطعة أصبحت من أشهر ألعاب الكلمات هذه الأيام وإن عملية توليدها يدويا صعب جدا و يستغرق الكثير من الوقت والطاقة و إن أتمتة عملياتها أثارت اهتمام الباحثين من حين لآخر , في هذا التقرير سنلاحظ مقتطف عن نتائج ما وصل إليه العلم في خوارزميات الكلمات المتقاطعة وسنوضح الهدف وأهمية هذه اللعبة و خوارزمياتها وكيف يمكنها إفادة الأبحاث الأخرى أو الأعمال الأخرى , ثم سنقدم الخوارزمية والعمليات المساعدة لها وما يمكن فعله لتطوير هذه الخوارزمية أو تطوير عمليات أخرى بالخوارزمية .

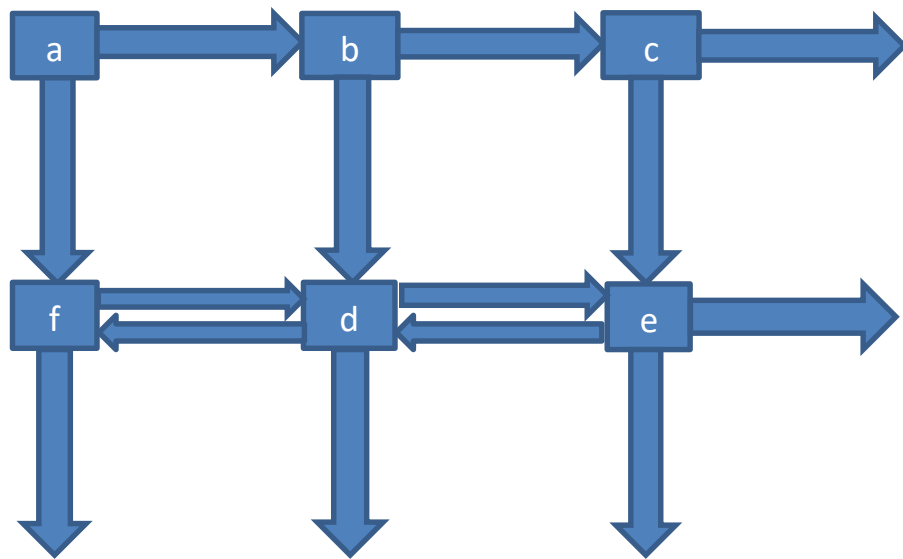
- في كل أسبوع نرى في الجريدة لعبة كلمات متقاطعة و نحن كمبرمجين ننظر إلى أهمية استثمار الوقت وأتمتة العمليات المتعبة والروتينية , فكم من الصعب توليد رقعة كلمات متقاطعة من حيث وضع كلمة ثم وضع كلمة أخرى تلائم الكلمات السابقة عموديا وأفقيا .

- سيقوم المشروع باستخدام قاعدة بيانات (قاموس) والذي يحوي على الكلمة وترجمتها (الكلمة هي اللغز و الترجمة هي السؤال) وسيقوم بمقاطعة الكلمات مع بعضها باستخدام خوارزميات سنقوم بشرحها ومن ثم انتظار من اللاعب إدخال الأحرف المناسبة وإظهار النتيجة بالأحرف الصحيحة و الخاطئة .

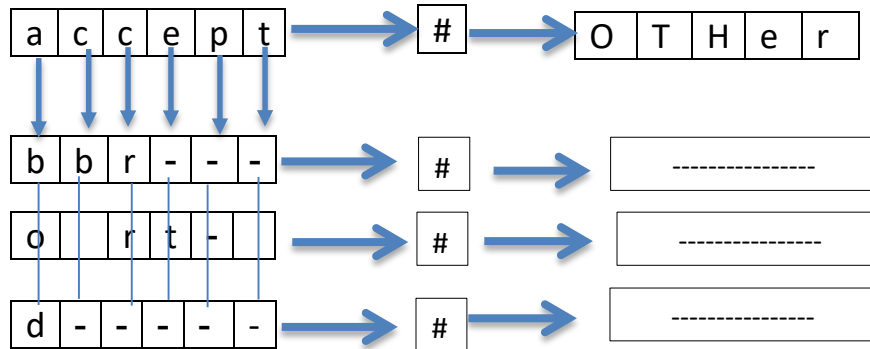
- نعم يمكن للإنسان أن يولد رقعة بسيطة لكن هل يمكنه توليد رقعة $40 * 40$ ببساطة ؟؟؟؟؟ خوارزمية الكلمات المتقاطعة يمكن استخدامها بتغيير القيود و بعض العمليات إلى الكثير من الأمور المفيدة مثل توليد برنامج إسبوع لجامعة أو توليد احتمالات لمخططات أو أمور تنظيمية

- في الواقع قد انتج العلم الكثير من خوارزميات الكلمات المتقاطعة كخوارزمية إضافة كلمة كلمة , حرف حرف والتي تقوم بوضع حرف و إيجاد الكلمات التي تحوي هذا الحرف وما هي احتمالات الأحرف التالية حسب الكلمات الناتجة وحسب تلائمها مع الأحرف المحيطة .

- حرف حرف :



- كلمة كلمة :



- والتي تقوم بوضع كلمة ما ثم وضع كلمة أخرى بحيث تلائم الكلمات الأخرى المحيطة و تتمتع هذه الخوارزمية بسرعتها عن الخوارزمية السابقة بسبب اختصار الحالات العودية (التراجع عن كلمة أقل من التراجع من حرف) و لذلك استخدمنا هذه الخوارزمية .
- وستقوم هذه الخوارزمية بتوليد جميع احتمالات أي رقعة كانت مهما كان بعدها اعتمادا على معالجة الحاسب وقدرته على الدخول باستدعاءات عودية عميقة وكثيرة و اختياره من بين عدد معين يمكنه معالجته من الكلمات .

شرح جميع الكلاسات

1. الكلاس الأساسي :

Class name	- The Game
Data member	<ul style="list-style-type: none"> -Char [][] The Game : مصفوفة الرقعة - Int Rows , colamns : أبعاد الرقعة -Int numberOfBlankBox : عدد المربعات السوداء - Int optimalSolution: عدد الرقع المثلى -Int foundedSolutio : عدد الرقع التي وجدت -List<char[][]>validAnswer : قائمة الرقع المثلى -List<list<string>>Ques1: قائمة الكلمات الأفقية بالرقع -List<string>oneGameQues1 : الكلمات الأفقية بأحد الرقع - List<list<string>>Ques2 : قائمة الكلمات العمودية بالرقع -List<string>oneGameQues2 : الكلمات الأفقية بأحد الرقع -Int[][]posArray1 : مصفوفة تقاطعات الأسئلة و الأجوبة الأفقية -Int[][]posArray2 : مصفوفة تقاطعات الاسئلة والأجوبة العمودية -List<integer>numberOfvalidsolution : عدد المربعات السود في الرقعة
Behavior	<ul style="list-style-type: none"> -The Game: باني بتهيئة الرقعة -printInfo : لطباعة رقعة -Addd : إضافة رقعة إلى الرقع : المقبولة - printlisrofvalidGame: طباعة الرقع المثلى -printQustion1 : طباعة الكلمات الأفقية - printQustuion2 : طباعة الكلمات العمودية - numberOfBlankBox : إيجاد عدد المربعات السوداء بالرقعة

2. الكلاس الثاني :

Class name	-The Qustion Object
Data member	الكلمة (الترجمة) : String the wordthatwewanttoadd عدد أحرف الكلمة : Number of letter ترجمات الكلمة (الاسئلة) : List<string>the translate عدد الأحرف بكل كلمة : Int[] number of word
Behavior	نسخ/ إضافة سؤال /تهيئة : The Qustion Object تصغير عدد الكلمات حسب الأحرف : Clear The number إعادة الكلمة : Get word إعادة ترجمات الكلمة : Get translate طباعة معلومات السؤال (الكلمة- الترجمات-عدد الأحرف) : printInfo

3. الكلاس الثالث :

Class name	- Blank Box
Data member	بعدي (موقع) المربع الأسود : Int x , y
Behavior	نسخ/ إضافة /تهيئة : Blank point تصفير قيمة المربع الأسود : Clear

4 – الكلاس الرابع :

Class name	-Propable char
Data member	<p>-Arraylist<Arraylist<character>>listolists:</p> <p>قائمة من الأحرف التي يمكن وضعها في كل المواقع مثال :</p> <p>- يوجد ثلاث مربعات فارغة يمكن وضع في المربع الأول a أو b أو c . ويمكن وضع في المربع الثاني f أو d . ويمكن وضع في المربع الثالث z .</p> <p>[0] → a , b , c ↓ [1] → d , f ↓ [2] → z</p> <p>Array List < character > singlelist [0] → a , b , c مثل :</p>
Behavior	-Propable char : تهئية/نسخ /إضافة

5 – الكلاس الخامس :

Class name	- NumericWord
Data member	<p>- String word : الكلمة</p> <p>- Int[] Number of word : عدد الكلمات التي تبدأ بحرف معين مثال : Number of word [3][2]=10 أي عدد الكلمات التي تحوي على 3 أحرف و تبدأ بال B = 10</p>
Behavior	<p>- Numeric word : تهئية /نسخ /إضافة</p> <p>-clear : تصفير المصفوفة</p>

6- الكلاس السادس :

Class name	Copy two Dimensional Array
------------	----------------------------

هو كلاس فقط لتعليب مصفوفة أرقام ثنائية لطباعتها بفايل .

الخوارزمية العامة :

```
Try ( The Game ) {  
    Find TheWordThatWeCanAdd[compatible word ]  
    Loop {  
        addTheword()  
        gowithNewTry()  
        backThelatestword()  
    }  
    If(checksolution()) {  
        Adddsolution()  
    }  
}
```



```

void tryy(TheGame GameObject){

    if(GameObject.numberofBlankBox < GameObject.foundedSolution
    && numberOfTry < ((numberOfGame*rows*column*5*4) / 1000)) {

        List<BlankPoint> BlankPoints = new ArrayList<BlankPoint>();

        int k = 0;

        TheQuestionObject[] ObjLetterToAdd = null;

        int NumberOfEmptyBox = 0;

        int[] Row_FirstColumns_LastColumns = {0, 0, 0};

        boolean returnBack;

        do {

            do {

                returnBack = false;

                Row_FirstColumns_LastColumns =
                FindThePlaceThatWeNeedToAdd_A_New_Word(GameObject);

                if (Row_FirstColumns_LastColumns[0] == 0) {

                    ObjLetterToAdd = new TheQuestionObject[0];

                    break;

                }

                NumberOfEmptyBox = numberOfBox(GameObject,
                Row_FirstColumns_LastColumns);

```

```

        if (returnBack =
checkNumberOfEmptyBox(NumberOfEmptyBox,
Row_FirstColumns_LastColumns, GameObject, BlankPoints))

        continue;

        PropableChar propableChar = GetPropableChar(GameObject,
Row_FirstColumns_LastColumns);

        if (returnBack = checkPropableChar(propableChar,
Row_FirstColumns_LastColumns, GameObject, BlankPoints))

        continue;

        do {

            if (GameObject.numberOfBlankBox >=
GameObject.foundedSolution) break;

            Row_FirstColumns_LastColumns[2] =
Row_FirstColumns_LastColumns[2] - k;

            if (Row_FirstColumns_LastColumns[2] <
Row_FirstColumns_LastColumns[1])

            break;

            NumberOfEmptyBox = NumberOfEmptyBox - k;

            if (returnBack =
checkNumberOfEmptyBox(NumberOfEmptyBox,
Row_FirstColumns_LastColumns, GameObject, BlankPoints))

            continue;

            int numberOfBlankBox = BlankPoints.size();

            ObjLetterToAdd = GetTheQuestionToAdd(propableChar,
NumberOfEmptyBox, BlankPoints, GameObject,
Row_FirstColumns_LastColumns);

```

```

        int numberOfBlankBox1 = BlankPoints.size();

        if(returnBack =
checkNumberOfBlankBox(numberOfBlankBox1, numberOfBlankBox))
break;

        boolean nothing = checkIfAllIsValid(ObjLetterToAdd);

        if (nothing) {

GameObject.TheGame[Row_FirstColumns_LastColumns[0]][Row_FirstColumns_LastColumns[2]] = '#';

        BlankPoints.add(new
BlankPoint(Row_FirstColumns_LastColumns[0],
Row_FirstColumns_LastColumns[2]));

        returnBack = true;

        break;

        }

propableChar.listOLists.remove(propableChar.listOLists.size() - 1);

        k++;

        } while (ObjLetterToAdd.length == 0);

    } while (returnBack);

    for (TheQuestionObject var : ObjLetterToAdd) {

        if (!var.getWord().equals("#")) {

            if (checkRepitedly(GameObject, var.getWord())) {

```

```

        System.out.println("here : " + var.getWord());

        AddWord(GameObject, var,
Row_FirstColumns_LastColumns, BlankPoints);

        tryy(GameObject);

        backTheLatestAnswer(GameObject,
Row_FirstColumns_LastColumns, BlankPoints);

    }

}

}

        numberOfTry++;

        System.out.println("numberOfTry = " + numberOfTry + "
(numberOfGame : " + numberOfGame + ") (rows : " + rows + ") (column : "
+ column + ") (((numberOfGame*rows*column*4) / 100) : " +
((numberOfGame*rows*column*4) / 100));

        if (checkIfThisAnAnswer(GameObject)) {

            System.out.println("valid : ");

            GameObject.PrinInfo();

            if (GameObject.ValidAswer.size() <
GameObject.OptimalSolution) {

                GameObject.addd(GameObject.TheGame);

            } else {

                GameObject.numberOfBlankBox++;

                optimalSolution(GameObject.TheGame, GameObject,
GameObject.numberOfBlankBox(GameObject.TheGame));

            }

        }

        int nn = Row_FirstColumns_LastColumns[2];

```

```

        int nnn = NumberOfEmptyBox;

        if (nnn < 3) {

            break;

        }

        if (nnn >= 3) {

GameObject.TheGame[Row_FirstColumns_LastColumns[0]][Row_FirstColumns_LastColumns[2]] = '#';

            BlankPoints.add(new
BlankPoint(Row_FirstColumns_LastColumns[0],
Row_FirstColumns_LastColumns[2]));

        }

        }while(GameObject.numberOfBlankBox <
GameObject.foundedSolution && numberOfTry <
((numberOfGame*rows*column*5*4) / 1000));

    }

}

```

أولا سنقوم بشرح التوابع المستخدمة في تابع ال Try ثم سنعرض الحالات التي تمت مناقشتها:

التابع الأول :

اسم التابع	-FindtheplacethatweNeedToAdd-A-New-word
البارامترات	The Game من Object
الخرج	مصفوفة int[] بالموقع الذي يجب وضع كلمة جديدة فيه مثل : السطر الثاني من المربع 1 إلى المربع 10 [2 , 1 , 10] السطر الثالث من المربع 3 إلى المربع 5 [3 , 3 , 5]
عمل التابع	-يوجد مصفوفة الموقع
الخوارزمية	<p>1-يمر على الرقعة طالما وصلنا إلى عنصر ليس #</p> <pre> 0 1 0 # # # # # 1 # \$ \$ \$ # </pre> <p>Stop → [1 , 1 , ?]</p> <p>2-يمر حتى يصل إلى عنصر #</p> <pre> 0 1 2 3 4 # # # # # # \$ \$ \$ # </pre> <p>Stop → [1 , 1 , 3]</p>

التابع الثاني :

اسم التابع	-numberOfBox
البارامترات	مصفوفة الموقع : -Int[] RowFirstColumns-lastColumns
الخرج	- Int : يمثل عدد المربعات الفارغ
عمل التابع	-يعد المربعات الفارغة
الخوارزمية	-طرح العنصر الثاني من الثالث

التابع الثالث :

اسم التابع	-Getrepable Char
البارامترات	-The Game GameObject -Int[] Row-firstColumns-LastColumns
الخرج	-يقوم بإرجاع أوبجكت يحوي على الأحرف الممكن وضعها في المربعات الفارغة
عمل التابع	-إيجاد الأحرف الممكن وضعها في المربعات الفارغة
الخوارزمية	1-إيجاد الكلمات العمودية التي تقع أعلى الكلمة باستدعاء التابع findTheprefixword ثم إيجاد الأحرف الممكنة حسب تلائماتها مع الكلمات العلوية باستدعاء التابع Getwordthatwecanuse

التابع الرابع :

اسم التابع	-findTheprefixword
البارامترات	-Int[]placeToAddin -The Game GameObject
الخرج	-مصفوفة سترينغ تحوي على الكلمات العمودية التي تقع اعلى المربعات الفارغة
عمل التابع	- يوجد الكلمات العمودية الموجودة اعلى المربعات الفارغة
الخوارزمية	

التابع الخامس :

اسم التابع	-GetwordThatwecanTOuse
البارامترات	الكلمات العلوية : prefix -String[]
الخرج	-يقوم بإرجاع الأحرف المحتمل وضعها
عمل التابع	-يقوم بالبحث عن الكلمات المشابهة لها و ماهي الأحرف التي وجدت والتي تعبر عن احتمال لإكمال الكلمة
الخوارزمية	1-إذا كانت الكلمة العمودية # أي يمكننا وضع أي حرف 2-نبحث عن كلمات مشابهة ونوجد احتمال الكلمة

التابع السادس :

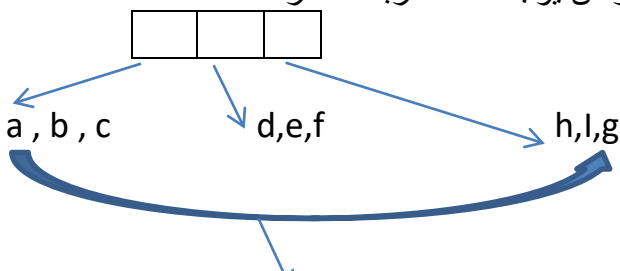
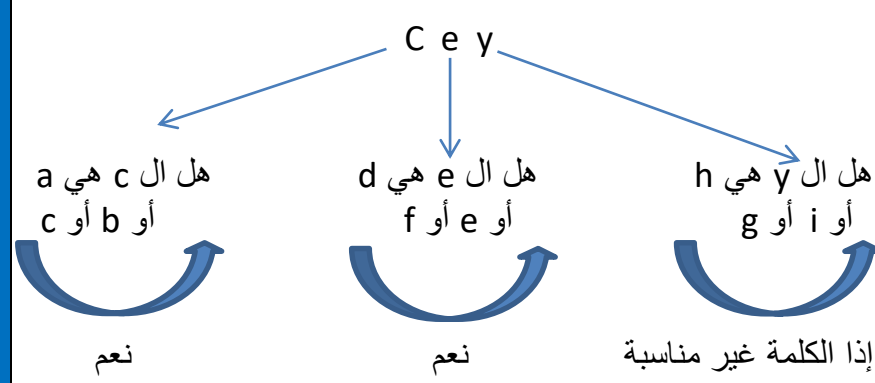
اسم التابع	-Compare First
البارامترات	-Char[] currentWord -Char[] ourWord
الخرج	- يرجع حرف
عمل التابع	- يرجع الحرف الممكن عند تشابه كلمتين
الخوارزمية	-ourWord abc -currentword abc de يرجع d لكن إذا كان Ourword abc currentword adc de يرجع # لأن $abc \neq adc$

التابع السابع :

اسم التابع	-GetTheQuestionToAdd
البارامترات	الأحرف المحتملة : propableChar Proablechar عدد المربعات الفارغة : int numberOfEmptyBox عدد المربعات السود(حاجز) : list<BlankPoint>BlankPoint الرقعة : TheGame GameObject مصفوفة الموقع : Int[]Row-firstColumns-lastColumns
الخرج	-يرجع أوبجكتات من الأسئلة والأجوبة الممكن وضعها بالرقعة

عمل التابع	-إيجاد الكلمات المناسبة و تحويلها إلى أو أوبجيكثات السؤال
الخوارزمية	1-إيجاد الكلمات المناسبة باستدعاء التابع : Findcompatibleword 2-البحث عن الكلمات باستدعاء التابع search لتحويل الكلمات إلى أوبجيكثات سؤال

التابع الثامن :

اسم التابع	-Findcompatibleword
البارامترات	-الأحرف المحتملة : propableChar Proablechar -عدد المربعات الفارغة : int numberOfEmptyBox -عدد المربعات السود(حاجز) : list<BlankPoint>BlankPoint -الرقعة : TheGame GameObject -مصفوفة الموقع : _Int[]Row-firstColumns-lastColumns
الخرج	-يرجع الكلمات الممكن استخدامها
عمل التابع	-إيجاد الكلمات الممكن وضعها بالرقعة التي تلائم الكلمات العمودية العلوية و عدد المربعات الفارغة
الخوارزمية	<p>بفرض يوجد ثلاث مربعات فارغة</p>  <p>الأحرف المحتملة</p> <p>تخرج جميع الكلمات بالقاموس التي تتألف من 3 أحرف مثلا :</p>  <p>نعم نعم إذا الكلمة غير مناسبة</p> <p>** يجب أن تكون جميع الأحرف نعم لتكون الكلمة مناسبة**</p>

التابع التاسع :

اسم التابع	-Search1
البارامترات	-اسم الكلمة
الخرج	-أوبجكت الكلمة إذا كانت موجودة
عمل التابع	- يبحث عن الكلمة عن طريق اسمها عن اسم الفايل الخاص ويخرج معلوماته
الخوارزمية	<p>-لنقوم بعملية بحث عن الكلمات بشكل سريع وكون الكلمات عددها كبير جدا (كونها قاموس حوالي 60000 كلمة) قمنا بمعالجة ملف القاموس بسحب الكلمات منه وأخذ ترجماتها وفتح فايل لكل كلمة بحيث :</p> <p>-اسم الفايل هو اسم الكلمة ويحوي الفايل على ترجمات الكلمة فعندما نريد البحث عن كلمة نفتح فايل باسم الكلمة إذا كان الفايل موجود ستكون ترجمتها داخل الفايل مع ذكر أن معالجة الفايل والقاموس باستخدام التابع Load –and-store</p>

التابع العاشر :

اسم التابع	-Load –and-store
عمل التابع	- يقوم بمعالجة الفايل (القاموس) لتحويله إلى فايلات لكل كلمة
الخوارزمية	<p>1- إيجاد الكلمة من الفايل (حسب هيكلية الفايل) 2- إيجاد ترجمات الكلمة ووضعهم في ليست 3- زيادة NumericWord 4- تخزين المصفوفات والمعلومات المسحوبة من الفايل بفايلات</p>

التابع الحادي عشر :

اسم التابع	-addword
	-The Game GameObject
	-The QuesTion Object TheQuesTion Object : الاوبجكت

البارامترات	الذي يجب إضافته مصفوفة الموقع : -InT [] Row-First column-Last columns وهي ليست للمربعات السود : -List<Blankpoint> Blankpoint
الخرج	-Void
عمل التابع	-يقوم بإضافة الكلمة إلى اندكسات الرقعة
الخوارزمية	1-إضافة الكلمة إلى اندكسات الرقعة 2-وضع # بعد الكلمة (يمثل نهاية الكلمة)

التابع الثاني عشر :

اسم التابع	- back The latest Answer
البارامترات	الرقعة : -The Game GameObject مصفوفة الموقع : -inT [] Row-First columns-Last columns قائمة المربعات السود : - LisT <Blankpoint>Blankpoint
عمل التابع	-التراجع عن آخر كلمة وضعناها
الخوارزمية	1-إعادة الكلمة التي وضعناها إلى \$ 2-إعادة المربع الأسود الذي أضفناه إلى \$

التابع الثالث عشر :

اسم التابع	-optimal solution
البارامترات	وهي اخر رقعة تم إضافتها : - char [][] LatestArray وهي الرقعة : -TheGame GameObject وهو محدد المربعات السود في آخر : -Int number of BlankBox رقعة
الخرج	-Void
عمل التابع	-إيجاد الرقعة الأمثل حيب عدد المربعات السود الأقل
الخوارزمية	-إذا كان عدد المربعات السود في اخر رقعة أقل من عدد المربعات السود في الرقع المثلى نقوم باستبدال هذه الرقعة مع اسوء رقعة من الرقع المثلى

التابع الرابع عشر :

اسم التابع	-Check NumberofEmpTyBox
البارامترات	<p>-int NumberofEmptyBox : وهي عدد المربعات الفارغة</p> <p>-int [] Row-firsTcolumn-Lastcolumns وهي مصفوفة الموقع</p> <p>-The Game GameObjecT : وهي الرقعة</p> <p>-LiT<Blank point> Blank point</p>
الخرج	<p>True : يجب إعادة معالجة الرقعة</p> <p>False : تابع</p>
عمل التابع	-يقوم بسد الفراغ وإعادة المعالجة
الخوارزمية	<p>إذا كان مثلاً : abc\$#</p> <p>أي يوجد مربع واحد فارغ نقوم بوضع مربع أسود مكانه (لا يوجد كلمة من حرف واحد)</p>

التابع الخامس عشر :

اسم التابع	-check propalbleChar
البارامترات	<p>-propableChar propableChar : وهي الاحرف المحتملة</p> <p>-inT[] Row –firstTcolumn –LasTcolumns : وهي مصفوفة الموقع</p> <p>-The Game GameObjecT : وهي الرقعة</p> <p>-List<Blankpoint>Blankpoint : وهي المربعات السوداء</p>
الخرج	<p>True : أعد المعالجة</p> <p>False : تابع</p>
الخوارزمية	-سد مربع لا يمكن وضع حرف به

التابع السادس عشر :

اسم التابع	-Check Repitidly
البارامترات	<p>The Game GameObjecT</p> <p>String current Word</p>

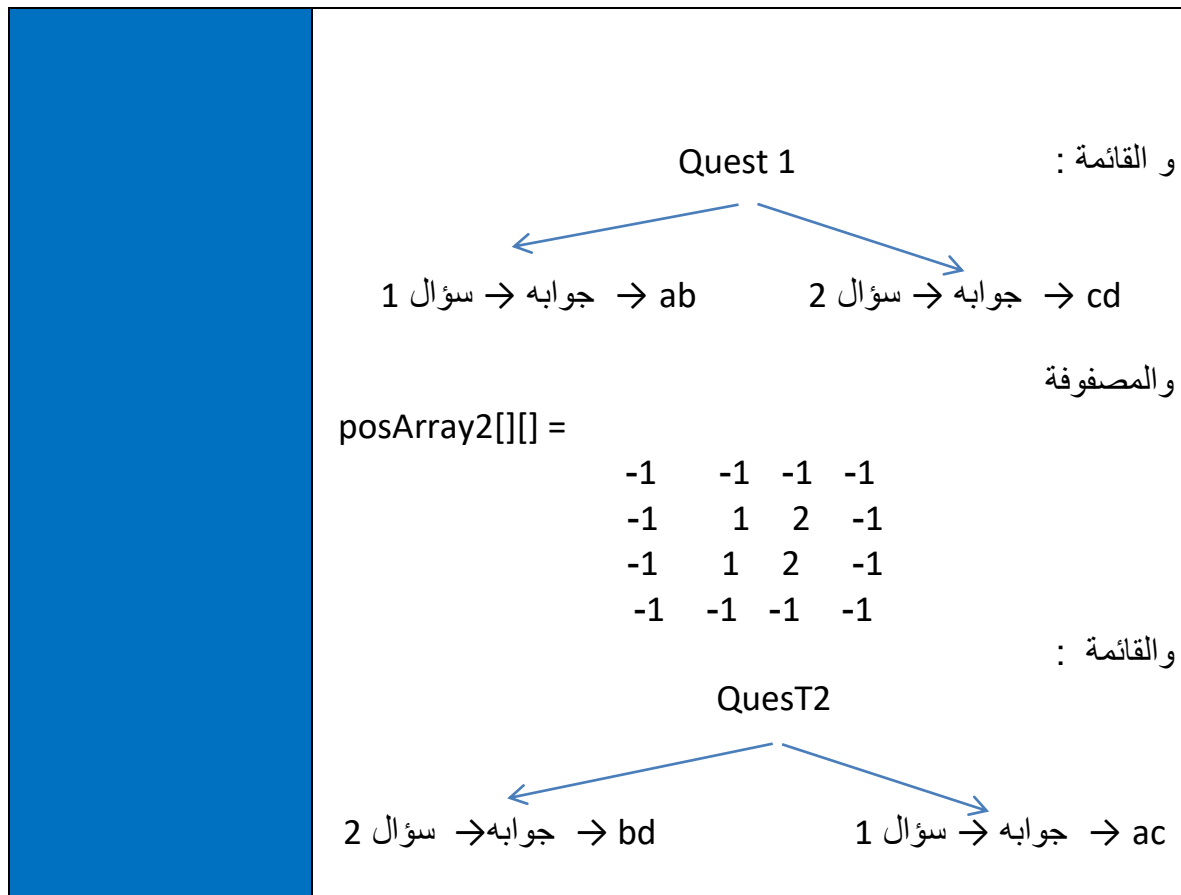
الخرج	True : الكلمة غير مكررة (أكمل) False : الكلمة مكررة (لا تكمل لاتضعها)
عمل التابع	- ضمان عدم وجود كلمات مكررة بالرقعة

التابع السابع عشر :

اسم التابع	-checkif This AnAnswer
البارامترات	-The Game GameObject وهي الرقعة
الخرج	True : وهي الرقعة كاملة False : الرقعة غير مكتملة
عمل التابع	-يضمن أن جميع المربعات قد تم تعبئتها

التابع الثامن عشر :

اسم التابع	- getQuesTion
البارامترات	-The Game GameObject
الخرج	-Void
عمل التابع	-إيجاد جميع الكلمات في الرقعة وإيجاد أسئلتها وربط مربعات الرقعة مع الاسئلة
الخوارزمية	<p>1-إيجاد الكلمات العمودية والافقية وأسئلتها</p> <p>2-تشكيل المصفوفتان posArray1[][], posArray[][] والقائمتين Quest1,Quest2</p> <p>كالتالي :</p> <pre> # # # # # a b # # c d # # # # # </pre> <p>يوجد لدينا الكلمات الافقية التالية cd , ab والعمودية التالية : ac,bd</p> <p>PosArray1[][]=</p> <pre> -1 -1 -1 -1 -1 1 1 -1 -1 2 2 -1 -1 -1 -1 -1 </pre> <p>ومنه:</p> <p>و القائمة:</p>



التابع التاسع عشر :

اسم التابع	-Draw
عمله	-رسم الرقعة

التابع العشرون :

اسم التابع	-compareTo
عمله	-لمقارنة أجوبة المستخدم مع الرقعة الصحيحة

التابع الحادي والعشرون :

اسم التابع	-color
عمله	-تقوم بالتالي باستخدام الرقعة و المصفوفات السابقة في التابع (رقم 18) # # # # 00 01 02 03

	<pre> # a b # 10 11 12 13 # c d # 20 21 22 23 # # # # 30 31 32 33 </pre> <p>إذا ضغطنا على العنصر 12 سنبحث عن العناصر التي تحوي القيمة 1 في posArray1 و القيمة 2 في posArray2 و منه :</p> <pre> # # # # # a b # # c d # # # # # </pre>
--	---

التابع الثاني والعشرون :

اسم التابع	-set index
عمله	<p>-يقوم بوضع رقمين (البعد [السطر والعمود] في id المربع مثلا : البعد بين 12 اي السطر 1 والعمود 2 فيكون الرقم id هو السطر*1000+ العمود اي 1002 وعندما نريد إرجاعه $\text{السطر} = \text{id} / 1000$ $\text{العمود} = \text{id} \% 1000$</p>

- التوابع التالية متشابهة وتقوم بطبع وقراءة للمعلومات إلى الفايلات المناسبة :

- read-Numeric-List
- Write-Numeric-List
- Write-Temp
- Write-TwoDimensional(Array)
- Write-oneDimensional(Array)
- Read-One –Dimenaional array
- Read-Two-Dimensional Array
- Read-Temp

و ختاماً سوف نقوم بمقارنة مشروعا بتطبيقات:

1- أحدهما عربي (وصلة) .

2- والاخر أجنبي (crossword) .

وصلة : الأسئلة والإجابات فيها (static) تم استخدامها وترتيبها مسبقاً و طريقة تقاطعاتها بسيطة حيث كل كلمة تتقاطع مع الأخرى من حرف أو حرفين...

crossword : تطبيق أجنبي يقوم بتوليد الكلمات المتقاطعة بطريقة سريعة و جيدة جداً () لكن أقصى بعد له هو $25 * 25$

- تطبيقنا يقوم بتوليد رقع الكلمات المتقاطعة بكلمات من قاعدة بيانات تحوي على 60000 كلمة ويمكنه توليد رقع تصل إلى $40 * 40$ و أكثر (أي لا حدود لبعد الرقعة) و لكن ذكرنا $40 * 40$ لأن الحاسب سيقع في عدم تحمل الاستدعاءات العودية بأبعاد أكثر من ذلك و هذا ما يمكن العمل عليه لتحسين عمل الخوارزمية (أي يمكن للخوارزمية توليد أي بعد لكن يجب مراعاة ال stack الذي يسجل العمليات العودية لتطوير العملية) .

- و يمكن أيضاً العمل على جعل الخوارزميات المستخدمة general أكثر لنستطيع استخدامها بتطبيقات أخرى و أيضاً استخدام اللعبة في أمور رائعة أخرى كتعليم لغة معينة (انكليزي) عن طريق نص قراءة و يكون ال test عنه عبارة عن لعبة بسيطة من الكلمات المتقاطعة .