مشروع الخوارزميات 1

إعداد الطلاب :

عمر حسن برهم فئة 14

عمر محمد بشار دالاتي فئة 14

علي عبد الله أسعد فئة 14

مشروع الخوارزميات :

المسألة الأولى :

الكود :

```
import java.util.*;

public class Q1 {
    public static void found(int
numbers[][], int rows, int column) {
        for (int i = 0; i < rows; i++)
            for (int j = 0; j < column; j++)
                if (numbers[i][j] == 0) {
                    System.out.println("The
object is in row " + (i) + " and column "
+ (j));
```

```
                break;
            }
    }

    public static void rec(int numbers[][],
int rows, int colomn, int i, int j) {
        int c = i + 1, s = j + 1;
        if (numbers[i][j] == 0) {
            System.out.println("The object is
in row " + (c - 1) + " and column " + (s -
1));
            numbers[i][j] = 1;
            return;
        }

        if (i + 1 >= rows && j + 1 ==
colomn - 1)
```

```
            rec(numbers, rows, colomn, i, s);
        else if (i + 1 == rows - 1 && j + 1
>= colomn)
            rec(numbers, rows, colomn, c, j);
        else  if (i + 1 >= rows || j + 1 >=
colomn)
            return;
        else {
            rec(numbers, rows, colomn, i, s);
            rec(numbers, rows, colomn, c, j);
        }
    }

    public static void rec1(int numbers[][],
int rows, int colomn, int i, int j) {
        int c = i + 1, s = j + 1;
        if (numbers[i][j] == 0) {
```

```
            System.out.println("The object is
in row " + (i) + " and column " + (j));
            numbers[i][j] = 1;
            return;
        }
        if (i + 1 >= rows && j + 1 ==
colomn - 1)
            rec(numbers, rows, colomn, i, s);
        else if (i + 1 == rows - 1 && j + 1
>= colomn)
            rec(numbers, rows, colomn, c, j);
        else if (i + 1 >= rows || j + 1 >=
colomn || numbers[i][j] == -1)
            return;
        else {
            rec1(numbers, rows, colomn, i,
s);
```

```
            rec1(numbers, rows, colomn, c,
j);
        }
    }

    public static void found1(int
numbers[][], int rows, int column) {
        if (numbers[0][0] == 0)
            System.out.println("The object is
in row 0 and column 0");
        else
            for (int k = 0; k < rows; k++)
                if (k + 1 != rows) {
                    if (numbers[k + 1][0] == -1)
{
                        for (int i = 0; i < column;
i++)
```

```
                        if (numbers[k][i] == 0)
{

System.out.println("The object is in row
" + (k) + " and column " + (i));
                            break;
                        }
                    }
                }else if ((k == rows - 1 &&
numbers[k][0] != -1))
                    for (int i = 0; i < column;
i++)
                        if (numbers[k][i] == 0) {
                            System.out.println("The
object is in row " + (k) + " and column "
+ (i));
                            break;
```

```
                    }
    }

    public static void rec2(int numbers[][],
int rows, int colomn, int x, int y, int i, int
j) {
        if (i < 0 || j < 0)
            return;
        numbers[i][j] = (x - i) + (y - j);
        rec2(numbers, rows, colomn, x, y, i
- 1, j);
        rec2(numbers, rows, colomn, x, y, i,
j - 1);
    }

    public static void main(String[] args) {
        int number[][] = {
```

```
                {1, 1, 1},
                {1, 1, 1},
                {1, 1, 0}
        };//change 3 to hom many column do you have in array
        System.out.println("Solution for 1 :");
        System.out.print("ITERATIVE ");
        found(number, number.length, 3);
        System.out.print("RECURSION ");
        rec(number, number.length, 3, 0, 0);
        System.out.println();
        number = new int[][]{
                {3, 2, 1},
                {2, 1, 0},
                {-1, -1, -1}
        };
```

```
        System.out.println("Solution for 2 part 1 :");
        System.out.print("iterative ");
        found1(number, number.length, 3);
        System.out.print("recursion ");
        rec1(number, number.length, 3, 0, 0);
        System.out.println();
        number = new int[][]{
                {-1, -1, -1},
                {-1, -1, -1},
                {-1, -1, -1}
        };
        int x = 2, y = 1;
        number[x][y] = 0;
        System.out.println("Solution for 2 part 2 :");
```

```
        rec2(number, number.length, 3, x, y,
x, y);
        for (int i = 0; i < number.length;
i++) {
            if (i > 0)
                System.out.println();
            for (int j = 0; j < 3; j++)
                System.out.print(number[i][j] +
" ");
        }
    }
}
```

OutPut:

Solution for 1 :

ITERATIVE The object is in row 2 and column 2

RECURSION The object is in row 2 and column 2

Solution for 2 part 1 :

iterative The object is in row 1 and column 2

recursion The object is in row 1 and column 2

Solution for 2 part 2 :

3 2 -1

2 1 -1

1 0 -1

الشرح :

الطلب الأول :

لدينا مصفوفة جميع عناصرها 1 نضع الغرض 0 في مكان معين في المصفوفة و نقوم بالبحث عن انديكس هذا عن طريق خوارزميتين

اولا الخوارزمية التكرارية : و هي عبارة عن حلقتين فور متداخلتين عندما نجد ان العنصر فيها هو 0 فاننا نطبع مكانه و هذه الخوارزمية تعقيدها

$$O(n*m)$$

ثانيا : الخوارزمية العودية : شرط التوقف فيها ان نصل الى خارج حدود المصفوفة

نبدأ من العنصر الأول و نتحقق من امكانية تقدمنا خطوة للاسفل و اليمين

اذا يمكننا فان التابع سيتدعي نفسه مرتين (مرة من أجل التقدم للاسفل خطوة واحدة و مرة من أجل تقدمه لليمين)

اذا وصلنا الى خارج حدود المصفوفة فاننا نغلق هذا الاستدعاء و نتحقق اذا كان يمكننا التقدم للاسفل دون التقدم لليمين فاننا نستدعي التابع من اجل هذه الخطوة و كذلك الامر ان كانت الخطوة لليمين

عندما نجد الغرض نجعل قيمته 1 و هذا دليل على اننا وجدناه و نطبع احداثياته و خذه الخوارزمية تعقيدها

$$O(2^{(n*m)})$$

الطاب الثاني :

الجزء الاول :

لدينا مصفوفة تحتوي على غرض قيمته 0 في انديكس ما الارقام التي على يساره -1 و اي ارقام تحته سطرها عبارة عن قيم جميعها -1 و باقي الخانات فيها ارقام تعبر عن مدى بعدها عن هذا الغرض المطلوب معرفة انديكس هذا الغرض

الخوارزمية التكرارية : و هي عبارة عن حلقتين فور متداخلتين تمشي على اول عنصر في اسطر
عندما اصل الى قيمة -1 فان الغرض المصفوفة موجود في السطر الذي خلفها نمشي على الاعمدة عندما نجد ان العنصر فيها هو 0 فاننا نطبع مكانه و اذا وصلنا الى اخر سطر و كان اول عنصر فيه لا -1 فحتما الغرض موجود في هذا السطر يساوي نمشي على الاعمدة عندما نجد ان العنصر فيها هو 0 فاننا نطبع مكانه و هذه الخوارزمية تعقيدها

O(n*m)

الخوارزمية العودية : نبدأ من اول عنصر في اول سطر اذا كان -1 فالمصفوفة المدخلة غير صحيحة و اذا كان 0 نطبع مكانه و نغير قيمته لكي لا نطبعها مرة اخرى و الا سنتحقق ان كان بامكاننا ان نخطو خطوة على اليمين من دون خارج حدود المصفوفة فاننا نستدعي التابع من اجلها و بنفس الطريقة من اجل الخطوة للاسفل

شرط التوقف ان تكون الخطوة للامام او للاسفل تخرجنا خارج الحدود المصفوفة او ان يكون قيمة الغرض الذي وصلنا اليه -1

الجزء الثاني :

لدينا مصفوفة جميع عناصرها -1 يضع الغرض في مكان محدد فيها

نقوم باستدعاء التابع المناسب و مهمته يقوم بتعبئة عناصر المصفوفة التي فوق و يمين الغرض بالقيمة

(x-i)+(y-j)

x and y : index of object

i : the row which are in

y : the column which are in

البداية دائما جميع هذه المتحولات هي انديكس الغرض ثم نقوم بانقاص عدد الاسطر التي نحن فيها بمقدار 1 و كذلك الامر بالنسبة للاعمدة و نسند هذه القيم لعناصر المصفوفة و هذه القيم تعبر عن مدى بعدنا عن الغرض و اخيرا نطبع المصفوفة التي حصلنا عليها و بما اننا استخدمنا خوارزمية عودية فتعقيدها

O(2^(n* m))

المسألة الثانية :

الكود :

```
import java.util.ArrayList;
import java.util.Arrays;

public class Q2 {
   public static void
autocom(ArrayList<String>words,String
aut)
   {
      for(int i=0;i<words.size();i++)
      {
         int y=0;
         for(int k=0;k<aut.length();k++)
         {
```

```
if(words.get(i).charAt(k)!=aut.charAt(k))
            {
               y=1;
               break;
            }
          if(y==0)

System.out.println(words.get(i));
       }
   }}
   public static void
check(ArrayList<String>words,String
check)
   {
      int j=0;
      for(int i=0;i< words.size();i++)
```

```
            if(check.equals(words.get(i)))
            {
               j=1;
               break;
            }
        if(j==0)
            System.out.println("This word
don't found in the dictionary \n");
        else
            System.out.println("This word
found in the dictionary \n");
    }
    public static void
add(ArrayList<String> words, String a[]) {
        for (String value : a) {
            boolean found = false;
```

```
            for (String word : words) {
                if (word.equals(value)) {
                    found = true;
                    break;
                }
            }
            if (!found) {
                words.add(value);
            }
        }
    }
    public static void main(String[] args){
        String
[]a={"String","float","Class","for"};
        String
[]b={"void","int","JAVA","JAVA"};
```

```
        ArrayList<String>words=new
ArrayList<>();
        for(int i=0;i<a.length;i++)
        words.add(a[i]);
        add(words,b);
        System.out.println("Solution 1 :");
        for(int i=0;i< words.size();i++)
            System.out.println(words.get(i));
        System.out.println("Solution 2 :");
      check(words,"JAVA");
        System.out.println("Solution 3 :");
    autocom(words,"f");
              }

}
```

OutPut:

Solution 1 :

String

float

Class

for

void

int

JAVA

Solution 2 :

This word found in the dictionary

Solution 3 :

float

for

الشرح :

لدينا مصفوفتين من نوع سترينغ الاولى تحوي كلمات المعجم الاساسية لبتي من اصل اللغة و الثانية تحوي المتحولات التي تريد اضافتهم للمعجم وarray list من نوع سترينغ

اولا : سيضيف المصفوفة الأولى الى array list ثم سيضيف الثانية لكن بشرط الا توجد كلمات مكررة

(اذا وجد كلمة من المصفوفة الثانية موجودة في array list لا يضيفها) وذلك باستخدام حلقتي فور

ثانيا : سيأخذ كلمة من المستخدم و سأتحقق اذا كانت موجودة في array list اذا كانت موجودة اطبع انها موجودة و اذا كانت غير موجودة اطبع انها غير موجودة و ذلك باستخدام حلقة فور

ثالثا : الاكمال التلقائي سأخد من المستخدم سترينغ تمثل بداية الكلمة التي يريد ان اكملها و سأتحقق من كل عنصر في array list

ساتحقق من كل حرف من الكلمة مع كل حرف من الكلمة التي اخذتها المعجم هل هما متساويان ام لا

اذا انتهت الكلمة و لم يجد اي حرف مختلف سوف يطبعها و الا لن يقوم بشيء

المسألة الثالثة :

الكود :

```
import java.util.*;
public class Q3 {
    public static int ans_p=0,ans_w=0;
    public static void  solution(int a[][],int
rows,int ir,int rs,int rw,int rp,int size,int
weight)
    {
        if(ir==rows)
```

```
        {
            if(ans_p<=rp)
            {
                ans_p = rp;
                if (ans_w <= rw)
                    ans_w = rw;
            }
        }
        else
        {
            int c=ir;

solution(a,rows,ir+1,rs,rw,rp,size,weight)
;
            if(rs+a[c][0]<=size &&
rw+a[c][1]<=weight)
```

```
solution(a,rows,ir+1,rs+a[c][0],rw+a[c][1]
,rp+a[c][2],size,weight);
        }}
    public static void main(String[] args) {
        int objects[][] = {
                {2,3,2},
                {2,3,5},
                {5,3,2},
                {10,10,10}
        };
        //size weight price
        //we have 3 rows and 3 fixed
column
      for (int i = 0; i < objects.length; i++) {
      if(i>0)
```

```
            System.out.println();
            for (int j = 0; j < 3; j++)
                System.out.print(objects[i][j] + "
"); }
        System.out.println();
        int size=10;
        int weight = 10;
        System.out.println("Capacity size
"+size);
        System.out.println("Capacity weight
"+weight);
        solution(objects, objects.length, 0,
0, 0, 0, size, weight);
        System.out.println("The Price is : "+
ans_p );
        System.out.println("The weight is :
```

```
"+ ans_w);
    }}
```

OutPut :

2 3 2

2 3 5

5 3 2

10 10 10

Capacity size 10

Capacity weight 10

The Price is : 10

The weight is : 1

الشرح :

لدينا مصفوفة من الاغراض التي تملك حجم و وزن و سعر و لدينا حقيبة لها وزن محدد و حجم محدد سيتدعي التابع الذي ةظيفته ان يأتي باكبر سعر و اكبر حجم و ألا يتجاوز وزن و حجم الحقيبة عن طريق استدعاء نفسه مرتين مرة لن ياخذ الغرض و مرة سيأخذه اذا كان وزن الغرض + وزن الاغراض التي معه لا تتجاوز وزن الحقيبة و حجم الغرض + حجم الاغراض التي معه لا تتجاوز حجم الحقيبة و عندما يصل لاخر عنصر سيتحقق ان كان حصل على اكثر من الذي حصل عليه سابقا (من استدعاءات اخرى) سيعدل السعر الذي حصل عليه سابقا للسعر الحالي و نفس الطريقة بالنسبة للوزن و لا يمكنه التحقق من الوزن الا اذا تأكد من حصوله على مبلغ اعلى من الذي حصل عليه سابقا