Stack

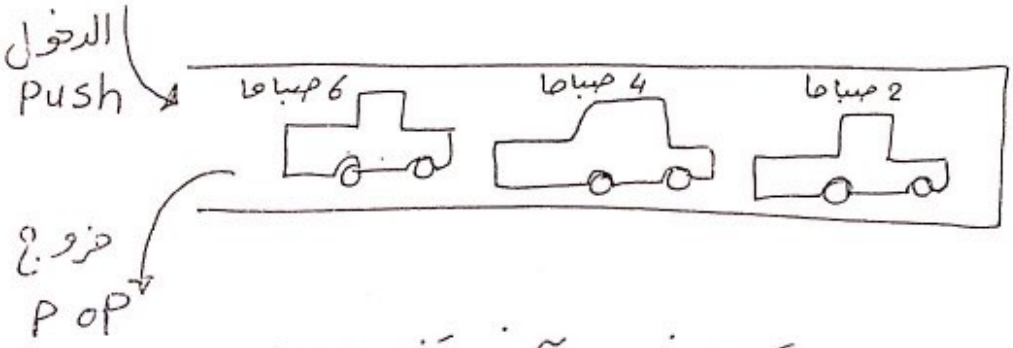
Posfix, Infix

م/ محمد فؤاد
0507979205

6

هو Linked List ولكن الحذف والاضافة من المقدمة

ويمكن تنفيذه ايضا باستخدام Array



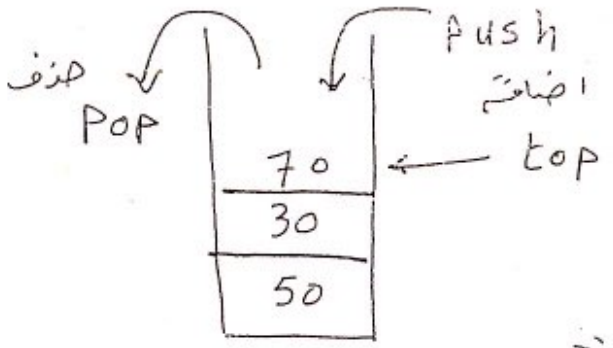
السيارة التى تدخل الأخيرة تخرج اولا

Last IN First out LIFO

Push(50)

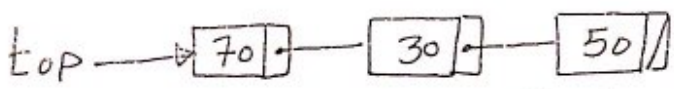
Push(30)

Push(70)



top يشير دائما الى آخر عنصر تمت اضافته

اذا كان top = null يكون Stack فارغ



اضافة Push هو نفس كود اضافة فى المقدمة LinkedList

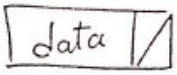
حذف Pop " " " حذف من المقدمة فى LinkedList

ADT Stack: Specification

Elements: The elements are of a variable type <Type>. In a linked implementation an element is placed in a node.

```
public class Node<T> extends Object {
        public T data;
        public Node<T> next;
        public Node () {
          data = null;  next = null; }
        public Node (T val) {
          data = val;  next = null; }
}
```

كلاس البنود السابقة



Structure: the elements are linearly arranged, and ordered according to the order of arrival, most recently arrived element called top.

Domain: the number of elements in the stack is bounded therefore the domain is finite. Type of elements: Stack

Operations: العمليات

All operations operate on a stack S.

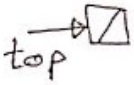
1. Method Push (Type e) — إضافة
   requires: Stack S is not full. input: Type e. — الاحدث
   results: Element e is added to the stack as its most recently added elements.
   output: none.

Push(X) — اضافة في المقدمة

2. Method Pop (Type e) — حذف
   requires: Stack S is not empty. input: — حذف من المقدمة
   results: the most recently arrived element in S is removed and its value assigned to e. output: Type e.

Pop()

3. Method Empty (boolean flag) — فارغ
   input: results: If Stack S is empty then flag is true, otherwise false. output: flag.



Empty

4. Method Full (boolean flag). — ممتلئ
   requires: input: .
   results: If S is full then Full is true, otherwise Full is false. output: flag.

ADT Stack (Linked Implementation) — التنفيذ باستخدام Linked List

```
public class LinkStack<T> {
  private Node<T> top;
  /* Creates a new instance of LinkStack */
  public LinkStack() {
    top = null; }
```

top لتعريف — لو نستخدم الكود

في البداية يكون Stack فارغ

head هو نفس top

Current لا يوجد

```
public boolean empty(){
    return top == null; }
```

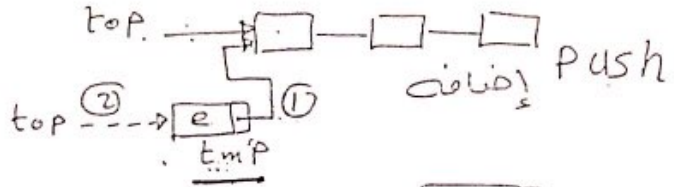
ترجع true إذا كان top يشير إلى null

```
public boolean full(){
    return false;
  }
```

عمرها ما تكون full

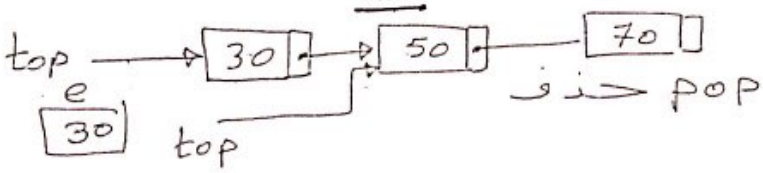
```
public void push(T e){
    Node<T> tmp = new Node(e);
    tmp.next = top;   (1)
    top = tmp;        (2)
  }
```

اضافة في المقدمة



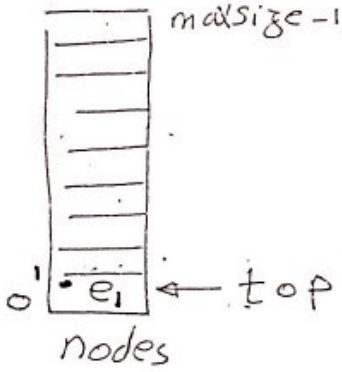

```
public T pop(){
        T e = top.data;
    top = top.next;
    return e;
  }
}
```

حذف من المقدمة



Stack: Array Implementation

التنفيذ على شكل Array

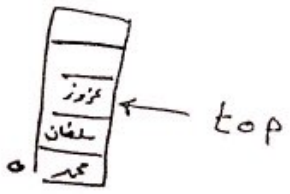



```
public class ArrayStack<T> {
  private int maxsize;
  private int top;
  private T[] nodes;
  /** Creates a new instance of ArrayStack */
  public ArrayStack(int n) {
    maxsize = n;
    top = -1;
    nodes = (T[]) new Object[n];
  }
```

n هو local variable الله يرحمه

كا ستغني

```
public boolean empty(){
    return top == -1;
  }
public boolean full(){
    return top == maxsize-1;
  }
```




```
public void push(T e){
    nodes[++top] = e;
  }
public T pop(){
    return nodes[top--];
  }
}
```

```
top++;
nodes[top] = e;
```

Push

```
T e = nodes[top];
top--;
return e;
```

Pop

م/ محمد فؤاد
0507979205

Postfix

Example: Convert from infix to postfix

3 + 8 * 2

حروف لدينا infix

Preceedance الاولويات

١- الاقواس ثم الاس

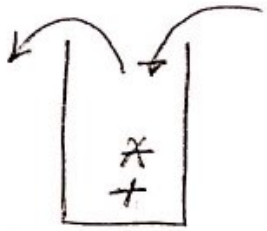
٢- الضرب * والقسمة / وباقي القسمة %

٣- الجمع + والطرح -

3 + 8 * 2 infix

عمليات

3 8 2 * + postfix



operators
عمليات فقط

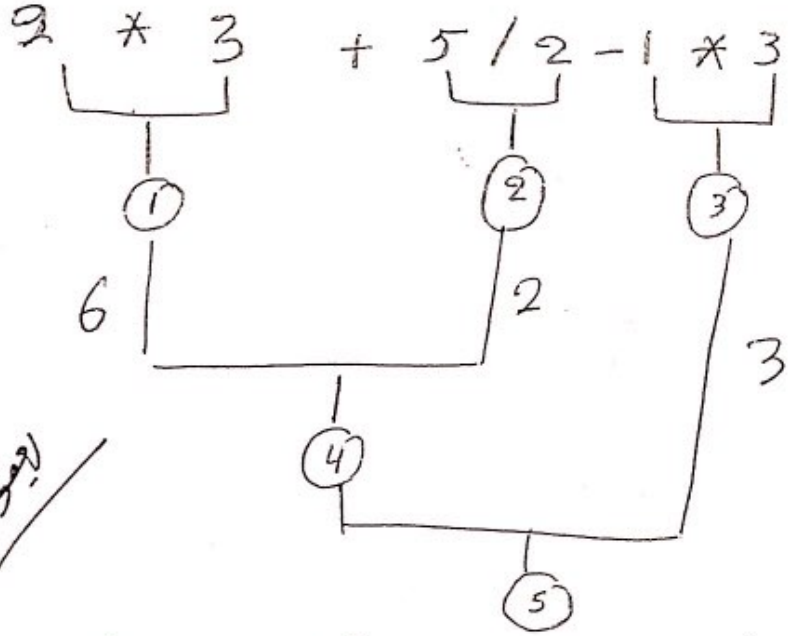
Convert from infix to postfix

2 * 3 + 5 / 2 − 1 * 3

⇓ Postfix

2 3 * 5 2 / + 1 3 * −

اوجد حلين



طريقة الحل

6 + 2 − 3 = 8 − 3 = 5

Convert form postfix to infix

2 3 * 5 2 / + 1 3 * -

*

6

/

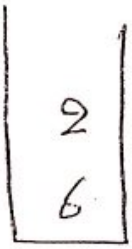
$5 / 2 = 2$

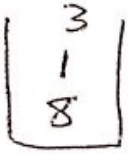
+

*

-

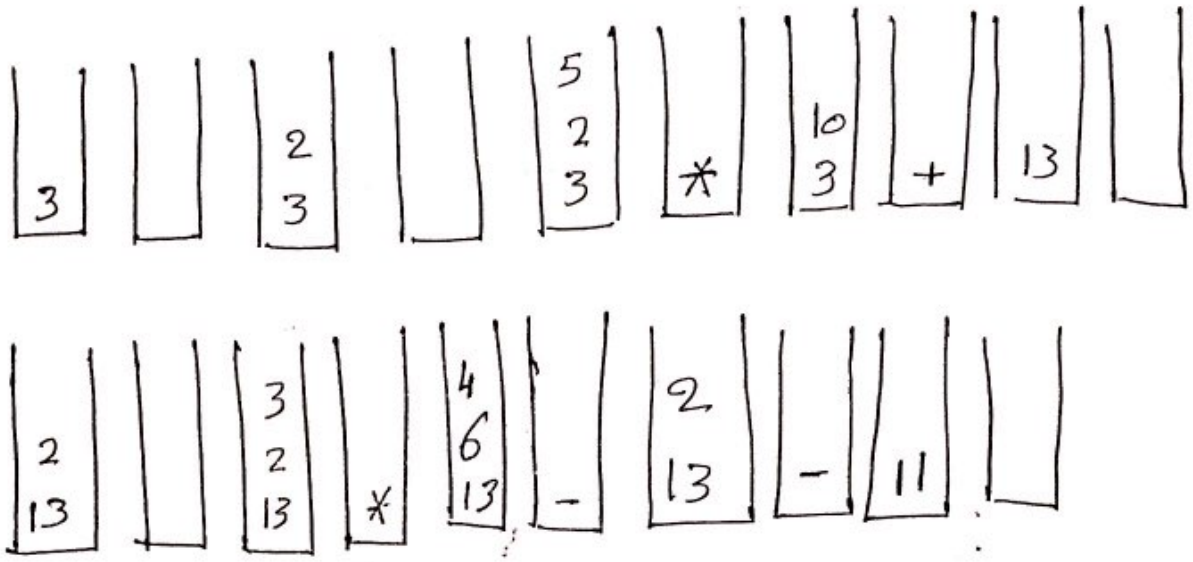
$8 - 3 = 5$





Evaluate using Stacks

3 2 5 * + 2 3 * 4 - -



2 * 5 = 10

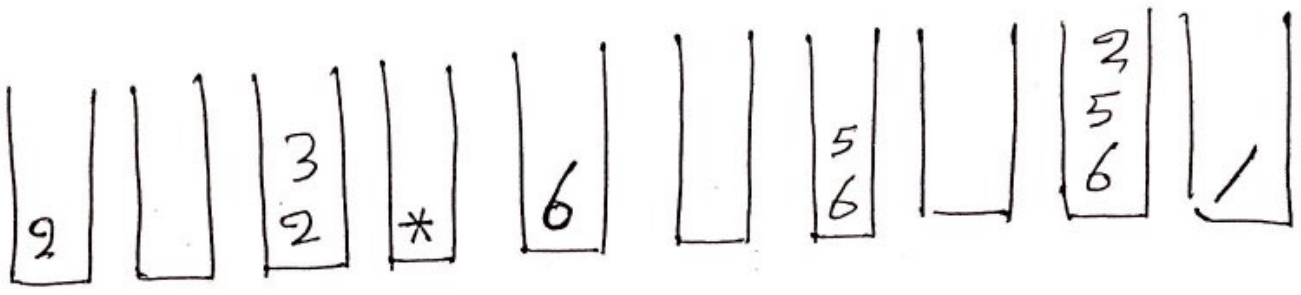
3 + 10 = 13

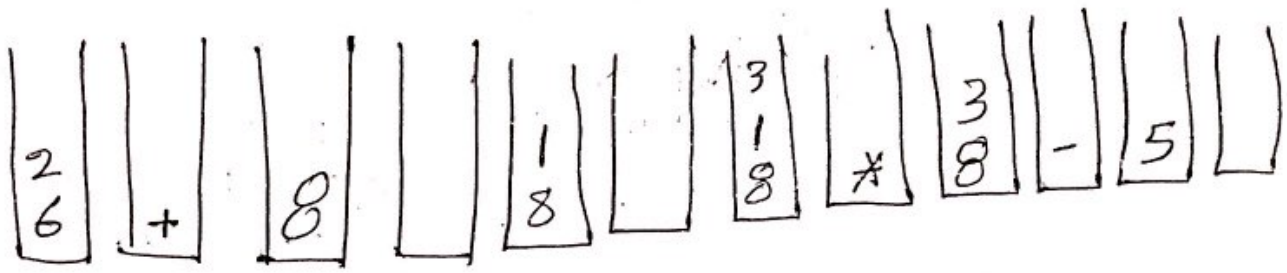
2 * 3 = 6

6 - 4 = 2

13 - 2 = 11

2 3 * 5 2 / + 1 3 * -





2 * 3 = 6

5 / 2 = 2

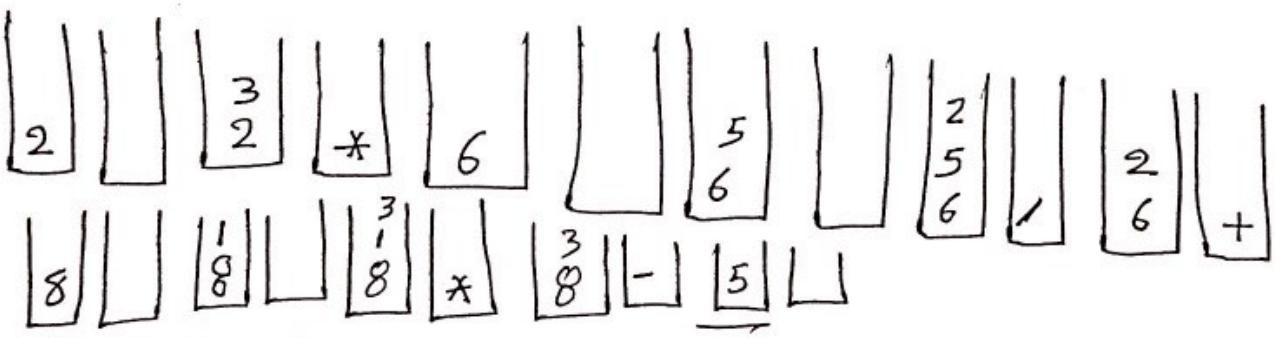
6 + 2 = 8

1 * 3 = 3

8 - 3

Evaluate using stacks

2 3 * 5 2 / + 1 3 * -



2 * 3 = 6

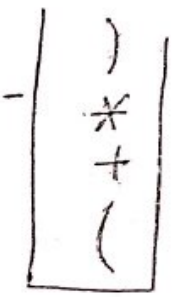
5 / 2 = 2

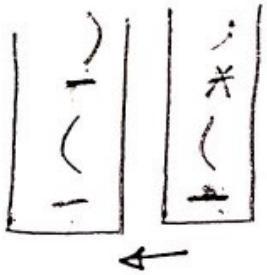
Convert from infix to postfix

$(3 + 2 * 5) - (2 * 3 - 4)$

3 2 5 * + 2 3 * 4 - -

post

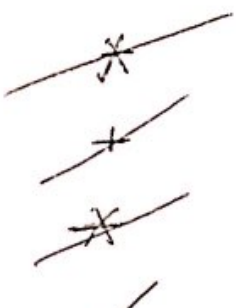


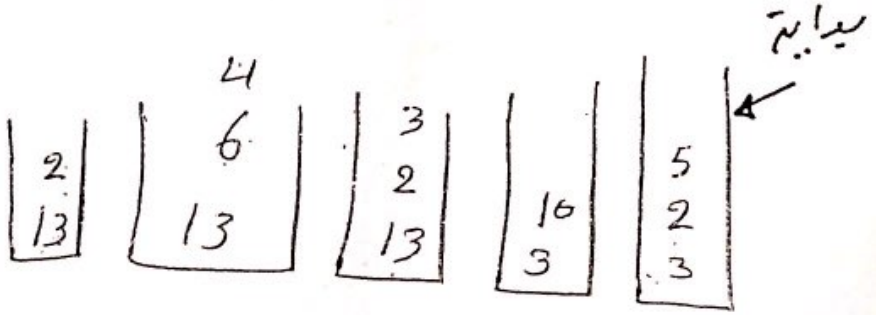


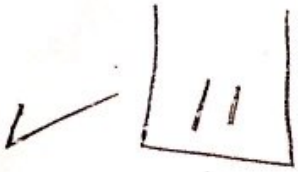
---

infix ىلإ post حول

3 2 5 * + 2 3 * 4 - -







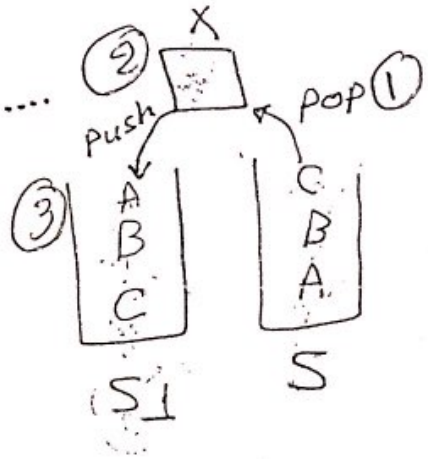
غاريم على Stack

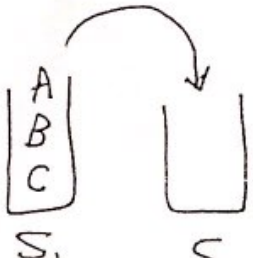
كتابة Write Method to print Stack

normal	Reverse
C	A
B	B
A	C

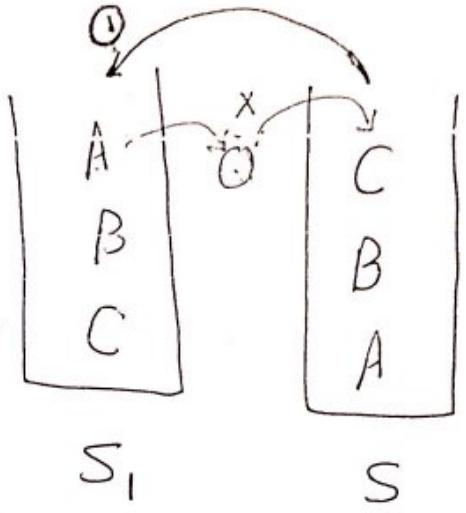
تريد طباعة C قبل A

```
public static void printNormal(Stack<T> S)
{
    Stack<T> S1 = new....   (new)
    T X;
    While ( !S.empty())
    {
        X = S.pop();        (1)
        System.out.println(X);  (2)
        S1.push(X);         (3)
    }
    While ( !S1.empty())
        S.push(S1.pop());
}
```





اللي تحوبو لتوتشو



لتزيد طباعة A قبل C

طباعة بالعكس

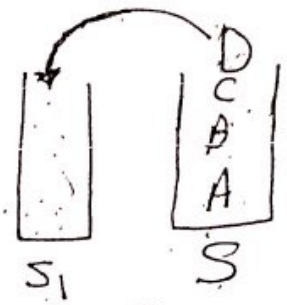
```
Public static PrintReverse(Stack<T> S)
{
    Stack<T> S1 = new Stack<T>();
    T X;
    while ( ! S.empty() )
    {
        S1.Push (S.PoP());
    }
    while ( ! S1.empty())
    {
        X = S1.PoP();
        S.o.p (X);
        S.Push (X);
    }
}
```

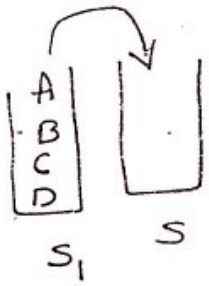
اللي ليوبو
ليوشو

Write static Method to find size of stack

حساب عدد العناصر الموجودة في stack

```
public static<T>int size (Stack<T> S)
{
    int C=0;
    Stack<T> S1=new ...
    while(!S.empty())
    {
        S1.push(S.Pop());
        C++;
    }
    while(!S1.empty())
    {
        S.push(S1.pop());
    }
    return C;
}
```

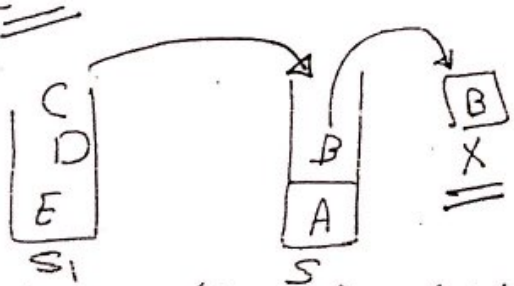




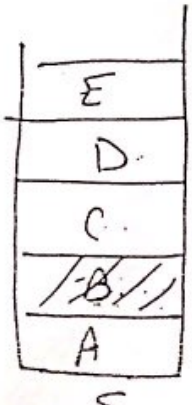
رجع العناصر

مطلوب حيذف اى عنصر من Stack

مطلوب حذف العنصر الرابع



حذف B وترجع S1 الى S

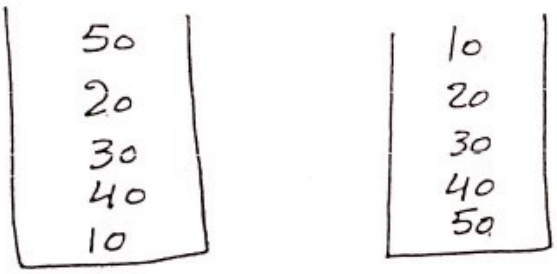


```
public static <T> void popSpecial(Stack<T> s, int n)   // كل كلى ... رقم العنصر
{
    Stack<T> S1 = new Stack<T>();
    for (int i = 0; i < n; i++)
    {
        S1.push(S.pop());
    }
    T X = S.pop();   // العنصر المطلوب
    // s.push(X);  {لو أبي ارجعه}
    while (!S1.empty())   // اجمع العناصر
        S.push(S1.pop());
    return X;
}
```

ايه الكلام ده

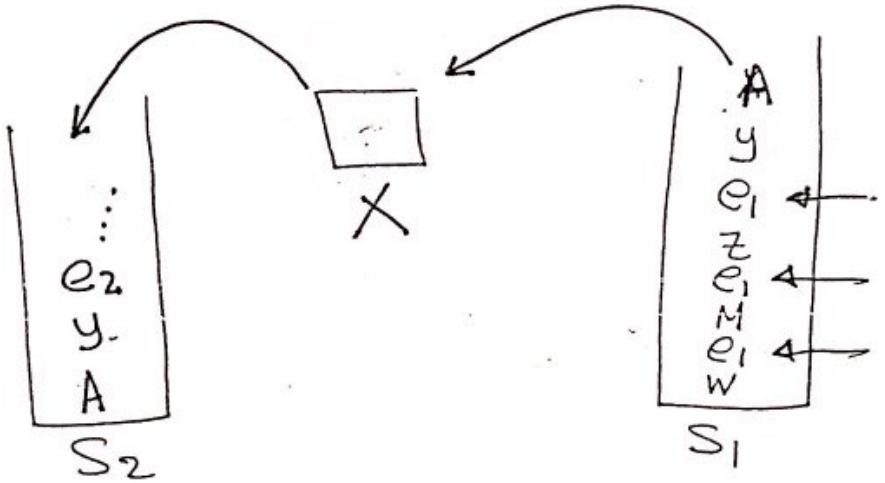
واجب

بدل اول عنصر مع آخر عنصر في Stack



سؤال هديه

استبدل كل قيمة e_1 بالقيمة e_2



* نقوم بعمل pop لاول عنصر ووضعه في مخزن X

اذا كانه $e_1 = X$ ثم بعمل push

لـ e_2 في S_2 والا ثم بعمل push X

* يتم تكرار الموضوع حتى يفرغ S_1

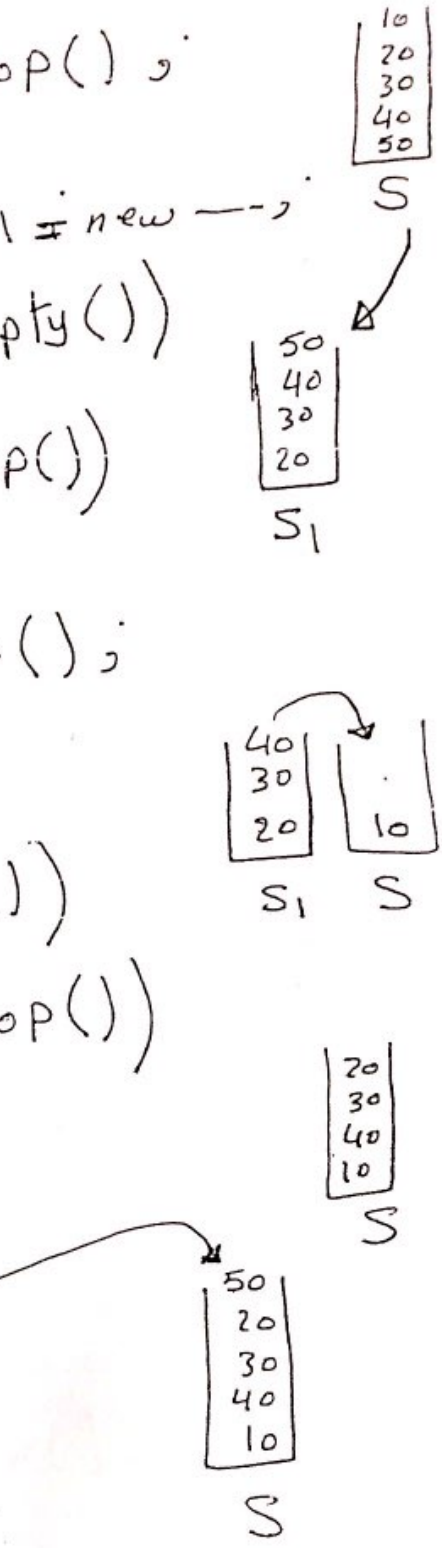
نقوم بإعادة S_2 الى S_1

```
Public static void SwapFirstLast(Stack<T> S)
{
    T X = S.pop();
    Stack<T> S1 = new ---;
    while (! S.empty())
    {
    S1.push (S.pop())
    }
    T y = S1.pop();

    S.push(X);
    while (! S1.empty())
    {
     S.push( S1.pop())
    }
    S.push(y);

}
```

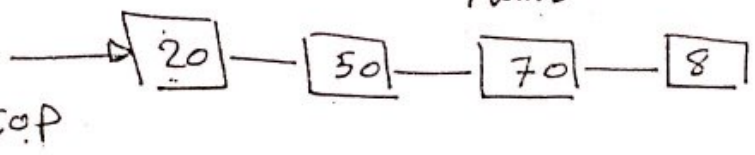
X = 10, y = 50



بدل اول عنصر آخر عنصر

```
public static<T>void replace( Stack<T> S1,
                               T e1, T e2 )
{
    Stack<T>S2 = new Stack<T>();
    T X;
    while ( ! S1.empty() )   // الاصلي
    {
        X = S1.pop();
        if (X.equals(e1))
            S2.push(e2);
        else
            S2.push(X);
    }
    while ( ! S2.empty())
        S1.push(S2.pop());   // راجع الفاضي
}
```

اكتب ميثود في ADT Member لترجع اول عنصر في Stack



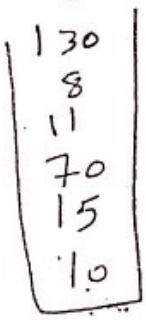
Write Method peek that return the first element of stak without delete it

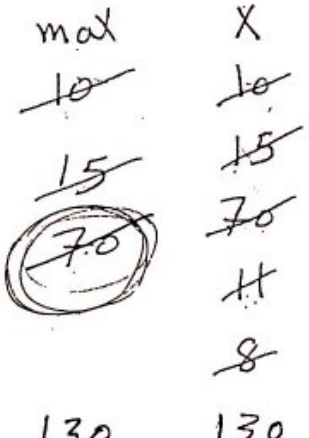
```
Member public T peek ( )
{
return top.data;
}
```

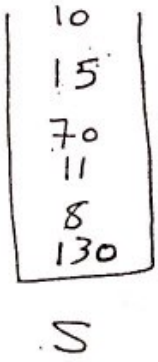
user

```
T x = S.pop();
S.push(x);
return x;
```

اكتب ميثود لايجاد Stack 3 max







S ج 3 S1 ج /

واجب ج / max

```
Public static T Max(LinkStack<T> S)
{
    Stak<T> S1 = new StackK<T>();
    T max, X;
    X = S.pop();        // اول عنصر
    max = X;            // اول عنصر
    S1.push(X);         // اول عنصر
    while (!S.empty())
    {
        X = S.pop();
        S1.push(X);
        if (X > max)
            max = X;
    }
    while (!S1.empty())
        S.push(S1.pop());
    return max;
}
```