Correction TP2

Exercice 3:

On utilize une structure de la forme:

```
struct pile {
    char stack[stack_size];
     int sp;
};
```

Version 1: **pile vide croissante**

Initialisation **sp=0**

```
 char pop()                          push(char c)
      return stack[--sp];                 stack[sp++] = c;
```

Version 2: **pile pleine croissante**

Initialisation **sp=-1**

```
 char pop()                          push(char c)
      return stack[sp--];                  stack[++sp] = c;
```

Version 3: **pile vide décroissante**

Initialisation **sp=stack_size**

```
 char pop()                          push(char c)
      return stack[++sp];                  stack[sp--] = c;
```

Version 3: **pile pleine décroissante**

Initialisation **sp=stack_size+1**

```
 char pop()                          push(char c)
      return stack[sp++];                  stack[--sp] = c;
```

Exercice 5:

CMP r0,r1 calcule r0-r1 et positionne les bits du cpsr en function du résultat.

a. (Z=1) ou N!=V

Z=1 veut dire que le résultat est nul, alors r0=r1

Si N!=V=1 (N=0 et V=1) comme N=0 le résultat est >=0 (bit 31 à 0) et comme V=1 les deux operands du calcul sont négatifs, r0<0 et (-r1)<0, c'est-à-dire r1>0 donc r1 > 0 > r0

Si N!=V=0 (N=1 et V=0) comme N=1 le résultat est négatif et V=0 indique qu'il n'y a pas d'overflow signé, alors on a r0-r1 < 0, donc r0<r1

Dans tous les cas on a toujours r0<=r1

b. Envoyez moi vos solutions, je corrigerai

c. (C=0) ou (Z=1)

Si C=0 alors r0-r1 = r0 + (max – r1 +1) <= max. donc r0-r1 <=-1, r0-r1 < 0, r0 < r1. Les calculs sont non signés car c'est le bit C qui est testé.

Si Z=1 le résultat est nul et r0=r1

Dans tous les cas on a r0 <= r1, non signé.

Remarquez qu'on a montré (C=0) ou (Z=1) => r0<=r1. Il faudrait aussi montrer l'implication inverse. La Remarque est aussi vraie pour les preuves du cours.