Corrigé Type du contrôle de TP AO :

Barème :

Exercice1 : chaque cas sur (0.25*3)=0.75 pt

Exercice 2 :

1) Pour la taille de chaque variable (0.25+0.25) pt. La taille du segment sur 0.25 pt

2) Pour chaque partie :

-le résultat de l'exécution des instructions (0.5 pt)

-la tâche sur 0.25 pt

3) la tâche du programme sur (0.5 pt).

Exercice 3 sur 3.75 pt

Exercice 4 :

1) Déclaration sur 0.5 pt

La saisie des deux chiffres : sur (0.5 pt)*2

La déduction du chiffre à partir de son code ASCII (0.5 pt)*2

L'addition et l'affectation dans RES sur 0.5 pt

2) sur 2 pts

Exercice 1 :

Cas	Erreurs ?	Erreurs Trouvées	Corrections
1	Oui	Dans la partie déclaration on déclare les variables et constantes. AX est un registre et non une variable ou constante.	A la place d'AX on utilise par exemple le nom d'une variable x : X DB 5
2	Oui	La valeur 1000 ne peut être stockée dans un octet mais elle a besoin de 2 octets.	S DW 1000
3	Oui	Pour que la fonction DOS numéro 9 puisse afficher le contenu d'une variable, cette dernière doit avoir un contenu qui termine avec '$'	Dans la partie DATA : Nombre DB ''Bonjour'','$'
4	Oui	L'octet [a+4] numéro 4 n'existe pas dans le segment de données	Dans la partie CODE, modifier l'offset ajouté à l'offset a : Add [a+i], 1 avec $0 \leq i \leq 3$
5	Oui	Les deux opérandes de mov n'ont pas la même taille : ax -> taille de 2 octets [x] -> taille de 1 octet	Dans la partie DATA : x DW 10

6	Oui	La multiplication mul[s] est de type 16 bits donc place le résultat dans (dx,ax). Le multiplication de 6000 par 10 peut tient sur 2 octets. Donc, al n'est pas suffisant, il faut récupérer le résultat de ax.	Partie DATA :
			S DW 0
			Res **DW** 0
			Partie CODE :
			Mov ax,6000
			Mul [s]
			Mov res,ax

7	Oui	Une boucle ne doit pas itérer et s'exécuter à l'infini. La valeur de BX ne change pas et elle fixée à 0.	A chaque itération de la boucle incrémenter la valeur de BX en utilisant par exemple : inc BX

Exercice 2 :

1)

Variable message-> taille=4 éléments *taille élément = 4*1 octet (DB -> 1 octet)
= 4 octets.
Variable input-> taille=6 éléments *taille élément = 6*1 octet (DB -> 1 octet)
= 6 octets.
Variable fin-> taille=1éléments *taille élément = 1*2 octet (DW -> 2 octets)
= 2 octets.

⇒ Taille du segment de données= somme des tailles des variables= 12 octets.

2)

Partie	Instruction	Résultat de l'instruction	Tâche de la partie
Partie1	2	Dx reçoit l'offset de input (c'est-à-dire 4)	La saisie d'une chaine contenant 2 ou 3 symboles (caractères).
	3	Ah reçoit 10	
	4	Appeler le DOS pour réaliser la fonction 10	
	5	al reçoit le contenu de l'octet (case) numéro 1 de input (taille de la chaine saisie)	
	6	Comparer le contenu de al avec 1	
	7	Si al>1 aller vers l'étiquette « suivant »	
	8	Sinon aller vers l'étiquette « saisie »	
Partie 2	10	bl reçoit le contenu de la case ou l'octet numéro 1 de input.	BX reçoit la taille de la saisie
	11	bh reçoit 0	
Partie 3	12	L'octet suivant la chaine saisie (contenant le retour chariot, c'est-à-dire 13) reçoit '.'	Placer après la chaine saisie la chaine **''.$''**
	13	L'octet suivant le '.' reçoit '$'	
Partie 4	14	Le 1er octet de input reçoit 13	Placer avant la chaine saisie 13 et 10
	15	Le 2ème octet de input reçoit 10	
Partie 5	19	ah reçoit 4ch	Retour au système d'exploitation DOS
	20	Appeler le DOS pour réaliser la fonction 4ch	

3)

Le programme saisit une chaine (contenant 2 ou 3 symboles) et affiche ''Bnjr'' suivi dans la ligne suivante de la chaine saisi, terminée par '.'

Exercice 3 :

```
    cmp [x],0        -> 0.5 pt
    ja else1         -> 0.5 pt
    mov z,0          -> 0.5 pt
    jmp sortir
else1 :
    cmp [x],10000    -> 0.5 pt
    ja else2         -> 0.5 pt
    mov z,1          -> 0.5 pt
    jmp sortir       -> les deux jmp sortir sur 0.25 pt

else2 :
   mov z,2           -> 0.5 pt
sortir :
```

Exercice 4 :

1)

```
cds segment para stack 'pile'
   db 256 dup (0)
cds ends
```

```
lds segment
chiffre1 db 0
chiffre2 db 0
res db 0
lds ends

lps segment
phrase proc far
      ASSUME CS :  LPS
      ASSUME DS : LDS
      ASSUME SS : CDS
      MOV AX , LDS
      MOV DS , AX

     Mov ah,1
     Int 21h
     Mov [chiffre1],al
     Mov ah,1
     Int 21h
     Mov [chiffre2],al
     Sub [chiffre1],48
     Sub [chiffre2],48
     Mov bl,0
     Add bl,[chiffre1]
     Add bl,[chiffre2]
    Mov [res],bl
     MOV AH,4CH
      INT 21H
phrase endp
lps ends
   end phrase
```

2) En s'inspirant de la solution de l'exercice 6 de la série de TP, on ajoute au programme précédant (question 1) le code suivant avant le retour au DOS (on a besoin d'une seule itération de la boucle réalisant la division) **:**

```
mov   al,[res]
mov ah,0          --> division sur 10   (0.5 pt)
mov cl,10
div cl

add al,48         --> convertir le quotient en ASCII  (0.25 pt)

mov dl,al
mov ah,2          --> Afficher le quotient (dizaines)   (0.5 pt)
int 21 h
add ah,48         --> convertir le reste en ASCII  (0.25 pt)
mov dl,ah         --> Afficher le reste (unités)  (0.5 pt)
mov ah,2
int 21 h
```