Université 08 Mai 1945 – Guelma

Faculté des MISM
Département des Mathématiques
1ère année Licence

Date: 14/05/2023
Durée : 2h00
Semestre 2

Corrigé type de l'examen d'Algorithmique et Structures de Données 2

Remarques:

- Toute solution correcte est acceptée.
- Les solutions en langage C sont également acceptées.

Exercice 1: (5 pts)

```
Const n=10;
Type Tab=Array[n] of Integer;
```

1) La fonction itérative `sommeIt`: (2 pts)

0.25	`Function sommeIt(T:Tab):Integer;`
0.25	`Var i,s:integer;`
	`Begin`
0.25	`s ← 0;`
0.25	`For i ← 0 To n-1 Do`
	`Begin`
0.5	`If T[i] mod 2 = 0 and T[i] mod 3 = 0 then`
0.25	`s ← s+T[i];`
	`End;`
0.25	`sommeIt ← s;`
	`End;`

2) La fonction récursive `sommeRec`: (3 pts)

0.25	`Function sommeRec(T:Tab;i:integer):integer;`
	`Begin`
0.75	`If i=10 then sommeRec ← 0`
0.25	`Else Begin`
0.5	`If T[i] mod 2 = 0 and T[i] mod 3 = 0 then`
0.75	`sommeRec ← T[i]+sommeRec(T,i+1)`
0.5	`Else sommeRec ← sommeRec(T,i+1);`
	`End;`
	`End;`

Exercice 2: (9 pts)

1) La structure de données: (0.5 pt)

```
Type List=^node;
     node=Record
          Begin
          val:integer;
          next:List;
          End;
```

2) La fonction `dernier_Urgent`: (3 pt)

0.25	`Function dernier_Urgent(L:List):List;`
0.25	`Var p:List;`
	`Begin`
0.5	`If L=Nil then dernier_Urgent ← Nil`
0.5	`Else if L^.val≠1 then dernier_Urgent ← Nil`
	`Else  begin`
0.25	`p ← L;`
1	`While p^.next≠Nil and p^.next^.val=1 Then`
	`p ← p^.next;`
0.25	`dernier_Urgent ← p;`
	`End;`
	`End;`

3) La fonction `ajouter_Patient`: **(5.5 pts)**

0.25	`Function ajouter_Patient(L:List,N:integer):List;`
0.5	`Var p,q,d:List;`
	`Begin`
0.5	`Allocate(p);`
	`p^.val ← N;`

0.75	`If L=Nil then`
	`  Begin`
	`  p^.next ← L;`
	`  L ← p;`
	`  End`

	`Else  Begin`
0.25	`  If N=1 then`
	`  Begin`
0.25	`  q ← dernier_Urgent3(L);`

0.75	`  If q=Nil then`
	`  Begin`
	`  p^.next ← L;`
	`  L ← p;`
	`  End`

0.75	`  Else  begin`
	`  p^.next ← q^.next;`
	`  q^.next ← p;`
	`  End`

	`  End`
	`  Else  Begin`
0.25	`  q ← L;`
0.5	`  While q^.next≠Nil Do` `  q ← q^.next;`
0.5	`  q^.next ← p;` `  p^.next ← Nil;`
	`  End;`
	`  End;`
0.25	`ajouter_Patient ← L;`
	`End;`

Exercice 3: **(6 pts)**

La structure de données : **(0.25 pt)**

```
Type Stack=^node;
     node=Record
          Begin
          val:integer;
          next:Stack;
          End;
```

1) La procédure `MinMax`: (3.25 pts)

0.25	`Procedure MinMax(P:Stack;Var min,max:integer);`
0.25	`Var P2:Stack;v:integer;`
	`Begin`
0.25	`initializeStack(P2);`
0.25	`pop(min,P);`
0.25	`max ← min;`
0.25	`push(min,P2);`
0.25	`While isStackEmpty(P)=False Do`
	`Begin`
0.25	`pop(v,P);`
0.25	`push(v,P2);`
0.25	`If v<min then min ← v`
0.25	`Else if v>max then max ← v;`
	`End;`
0.5	`While isStackEmpty(P2)=False Do`
	`Begin`
	`pop(v,P2);`
	`push(v,P);`
	`End;`
	`End;`

2) La procédure `RetirerMinMax`: (2.5 pts)

0.25	`Procedure RetirerMinMax(Var P:Stack);`
0.25	`Var P2:Stack;v,min,max:integer;`
	`Begin`
0.25	`initializeStack(P2);`
0.25	`MinMax(P,min,max);`
0.25	`While isStackEmpty(P)=False Do`
	`Begin`
0.25	`pop(v,P);`
0.5	`If v≠min and v≠max then push(v,P2);`
	`End;`
0.5	`While isStackEmpty(P2)=False Do`
	`Begin`
	`pop(v,P2);`
	`push(v,P);`
	`End;`
	`End;`