A. Tri par sélection :

☒ **Analyse du Programme principal** :

NOM : Sélection
Résultat=PROC Afficher(T, n)
PROC Tri_Selction(T, n)
PROC Remplir(T,n)
PROC Saisir(n)
FIN Sélection

T.D.N.T

Type
tab= tableau de 50 entiers

TDO.Globaux:

Objet	Nature/type	Rôle
n	Var/entier	La taille
T	Var/tab	tableau à trier
Saisir,Remplir, Tri_Selection,Afficher	procedure	

Analyse de la procédure Saisir :
DEF PROC Saisir (var n :entier)
Résultat= [ ]
Répéter
n=donnée("saisir la taille d'un tableau=")
Jusqu'à n>0
Fin Saisir

Analyse de la procédure Remplir :
DEF PROC Remplir **(var T :tab; n :entier)**
Résultat= [ ]
Pour i de 1 à n **faire**
T[i]=donnée("T[",i,"]=")
FinPour
Fin Remplir

TDO.Locaux:

Objet	Nature/type	Rôle
i	Var/entier	compteur

Analyse de la procédure Tri_Selection:
DEF PROC Tri_Selection (var T :tab ;n :entier)
Résultat= [ ] **Pour i de 1 à n-1 faire**
[pos_min←i] **Pour** j **d**e i+1 **à** n **faire**
[ ] **Si** (T[j] < T[pos_min])**Alors**
pos_min←j
Finsi
FinPour
[ ] **Si** (pos_min < > i)**Alors**
PROC Permute(T[i], T[pos_min])
Finsi
FinPour
FIN Tri_Selection

☒ **Algorithme du Programme principal** :
0) **Début Sélection**
1) **PROC Saisir(n)**
2) **PROC Remplir(T,n)**
3) **PROC Tri_Selction(T, n)**
4) **PROC Afficher(T, n)**
5) **Fin Sélection**

☒ **Algorithme de la procédure Saisir** :
0) **DEF PROC Saisir (var n :entier)**
1) [ ] **Répéter**
Ecrire("saisir la taille d'un tableau=")
Lire(n)
Jusqu'à n>0
2) **Fin Saisir**

☒ **Algorithme de la procédure Remplir** :
0) **DEF PROC** Remplir **(var T :tab; n :entier)**
1) [ ] **Pour** i de 1 à n **faire**
Ecrire("T[",i,"]="),Lire(T[i])
FinPour
2) **Fin Remplir**

☒ **Algorithme de la procédure Tri_Selection**:
0) **DEF PROC Tri_Selection (Var T :tab;n :entier)**
1) [ ] **Pour i de 1 à n-1 faire**
[pos_min←i] **Pour** j **d**e i+1 **à** n **faire**
[ ] **Si** (T[j] < T[pos_min])**Alors**
pos_min←j
Finsi
FinPour
[ ] **Si** (pos_min < > i)**Alors**
PROC Permute(T[i], T[pos_min])
Finsi
FinPour
2) **FIN Tri_Selection**

TDO.Locaux:

Objet	Nature/type	Rôle
i,j	Var/entier	compteur
pos_min	Var/entier	Position de la petite valeur
Permute	procédure	

Analyse de la procédure Permute :

DEF PROC Permute **(var a,b :entier)**

Résultat= aux ←a
a ←b
b←aux

FIN Permute

TDO.Locaux:

Objet	Nature/type	Rôle
aux	Var/entier	Variable intermédiaire

Analyse de la procédure Afficher:

DEF PROC Afficher **(T :tab ;n :entier)**

Résultat= [ ] **Pour** i de 1 à n **faire**
Ecrire(T[i],"|")
FinPour

Fin Afficher

TDO.Locaux:

Objet	Nature/type	Rôle
i	Var/entier	compteur

☒ **Algorithme de la procédure Permute** :

0) DEF PROC Permute **(var a,b :entier)**
1) aux ←a
2) a ←b
3) b←aux
4) FIN Permute

Algorithme de la procédure Afficher:

0) DEF PROC Afficher **(T :tab ;n :entier)**
1) [ ] **Pour** i de 1 à n **faire**
Ecrire(T[i],"|")
FinPour
2) Fin Afficher

Traduction Pascal :

```
program Selection;
uses wincrt;
type tab=array[1..50] of integer;
var n:integer;
   t:tab;

procedure Saisir(var n:integer);
begin
repeat
     write('saisir la taille d''un tableau= ');
     readln(n);
until n>0;
end;

procedure Remplir(var t:tab;n:integer);
var i:integer;
begin
for i:=1 to n do
 begin
  write('t[',i,']=');
  readln(t[i]);
end;
end;

procedure Tri_Selection(var t:tab;n:integer);
var i,j,posmin:integer;

       procedure permute(var a,b:integer);
       var aux:integer;
       begin
         aux:=a;
         a:=b;
         b:=aux;
 end;
begin
for i:=1 to n-1 do
 begin
  posmin:=i;
   for j:=i+1 to n do
    begin
      if(t[j]<t[posmin])then
        posmin:=j;
    end;
  if posmin <>i then
     permute(t[i],t[posmin]);
 end;
end;

procedure Afficher(t:tab;n:integer);
var i:integer;
begin
for i:=1 to n do
 begin
  write(t[i],' | ');
 end;
end;
{********** programme principal********}
begin
 Saisir(n);
 Remplir(t,n);
 Tri_Selection(t,n);
 Afficher(t,n);
end.
```

B. Tri à Bulles :

☒ **Analyse de la procédure Tri_Bulles**:

DEF PROC Tri_Bulles (var T :tab ; n :entier)
Résultat= [ ]
Répéter
[Echange←faux] **Pour** i de 1 à n-1 **faire**
[ ] **Si** (T[i] > T[i+1])**Alors**
PROC Permute(T[i], T[i+1])
Echange ← vrai
FinSi
FinPour
n←n-1
Jusqu'à (Echange = Faux) ou (n=1)
Fin Tri_Bulles

☒ **Algorithme de la procédure Tri_Bulles**:

0) DEF PROC Tri_Bulles (var T :tab ; n :entier)
1) Répéter
[Echange←faux] **Pour** i de 1 à n-1 **faire**
[ ] **Si** (T[i] > T[i+1])**Alors**
PROC Permute(T[i], T[i+1])
Echange ← vrai
FinSi
FinPour
n←n-1
Jusqu'à (Echange = Faux) ou (n=1)
2) Fin Tri_Bulles

TDO.Locaux:

Objet	Nature/type	Rôle
i	Var/entier	compteur
Echange	Var/booléen	Variable booléenne
Permute	procédure	

Traduction Pascal :

```
program Bulles;
uses wincrt;
type tab=array[1..50] of integer;
var  n:integer ;     t:tab;

procedure Saisir(var n:integer);
begin
repeat
     write('saisir la taille d''un tableau= ');
     readln(n);
until n>0;
end;

procedure Remplir(var t:tab;n:integer);
var i:integer;
begin
for i:=1 to n do
 begin
  write('t[',i,']='); readln(t[i]);
end;
end;

procedure Tri_Bulles(var t:tab;n:integer);
var i:integer;echange:boolean;
          procedure permute(var a,b:integer);
         var aux:integer;
         begin
           aux:=a;
           a:=b;
           b:=aux;
         end;
begin
repeat
 echange:=false;
 for i:=1 to n-1 do
     if(t[i]>t[i+1])then
      begin
       permute(t[i],t[i+1]);
       echange:=true;
      end;
n:=n-1;
until(echange=false) or (n=1);
end;

procedure Afficher(t:tab;n:integer);
var i:integer;
begin
for i:=1 to n do
 begin
  write(t[i],' | ');
 end;
end;
begin
 Saisir(n);
 Remplir(t,n);
 Tri_Bulles(t,n);
 Afficher(t,n);
end.
```

C. Tri par Insertion :

Analyse de la procédure Tri_Insertion :

```
DEF PROC Tri_Insertion (var T :tab ;n :entier)
Résultat= [ ] Pour i de 2 à n faire
            [x←T[i], j←i ]
    Tant que ( j >1) et ( T[j-1] > x ) faire
        T[j ]← T[j-1]
        j ← j - 1
   FinTantQue
  T[j ] ←x
FinPour
Fin Tri_Insertion
```

☒ Algorithme de la procédure Tri_Insertion:

```
0) DEF PROC Tri_Insertion (var T :tab ;n :entier)
1) [ ] Pour i de 2 à n faire
            [x←T[i], j←i ]
     Tant que ( j >1) et ( T[j-1] > x ) faire
        T[j ]← T[j-1]
        j ← j - 1
     FinTantQue
     T[j ] ←x
   FinPour
2) Fin Tri_Insertion
```

TDO.Locaux:

Objet	Nature/type	Rôle
i,j	Var/entier	compteur
x	Var/entier	

Traduction Pascal :

```
program Insertion;
uses wincrt;
type tab=array[1..50] of integer;
var n:integer;
   t:tab;

procedure Saisir(var n:integer);
begin
repeat
    write('saisir la taille d''un tableau= ');
    readln(n);
until n>0;
end;

procedure Remplir(var t:tab;n:integer);
var i:integer;
begin
for i:=1 to n do
 begin
  write('t[',i,']=');
  readln(t[i]);
end;
end;
```

```
procedure Tri_Insertion(var t:tab;n:integer);
var i,j,x:integer;
begin
    for i:=2 to n do
     begin
       x:=t[i];
       j:=i;
        while (j>1) and (t[j-1]>x) do
          begin
            t[j]:=t[j-1];
            j:=j-1;
          end;
      t[j]:=x;
     end;
end;

procedure Afficher(t:tab;n:integer);
var i:integer;
begin
for i:=1 to n do
 begin
  write(t[i],' | ');
 end;
end;
{********** programme principal**************}
begin
 Saisir(n);
 Remplir(t,n);
 Tri_Insertion(t,n);
 Afficher(t,n);
end.
```

II. Les recherches:

1. La recherche séquentielle :

☒ **Analyse de la fonction Recherche:**

```
DEF FN rech_seq (T:TAB ;n,v:entier): Booléen
Résultat=rech_seq←trouve
  [i←0,trouve←faux]Répéter
                        i←i+1
                        si (T[i]=v) alors
                              trouve←vrai
                        finsi
                  jusqu'a (trouve=vrai) ou (i=n)
Fin rech_seq
```

T.D.O.Locaux:

Objet	Nature/type	Rôle
i	Var/entier	Compteur
trouve	Var /booléen	Le résultat de la recherche

☒ **Algorithme de la fonction recherche:**

```
0) DEF FN rech_seq (T:tab ;n,v:entier): Booléen
1) [i←0,trouve←faux]Répéter
                        i←i+1
                        si (T[i]=v) alors
                              trouve←vrai
                        finsi
                  jusqu'a (trouve=vrai) ou (i=n)
2) rech_seq←trouve
3) Fin rech_seq
```

☒ **En Pascal:**

```
function rech_seq(T:tab;n,v:integer):boolean;
var i:integer; trouve :boolean ;
begin
      i:=0; trouve:=false;
         repeat
          i:=i+1;
          if(T[i] =v) then
              trouve :=true ;
        until (trouve=true)or (i=n)
 rech_seq:=trouve;
end;
```

2. La recherche dichotomique :

Remarque:

La recherche dichotomique s'applique sur un tableau trié.

☒ **Analyse de la fonction rech_dich:**

```
DEF FN rech_dich(T:tab ;n,v:entier): Booléen
Résultat=rech_dich←trouve
        [a←1, b←n, trouve←faux]
            répéter
                m← (a+b) DIV 2
                 si T[m]=v alors
                      trouve←vrai
                 sinon si T[m]>v alors
                                b←m-1
                 sinon
                            a← m+1
                 finsi
            jusqu'a (trouve=vrai) OU (a>b)
FIN rech_dich
```

T.D.O.Locaux:

Objet	Nature/type	Rôle
i	Var/entier	Compteur
trouve	Var /booléen	Le résultat de la recherche
a	Var/entier	Le début de l'intervalle de recherche
b	Var/entier	La fin de l'intervalle de recherche
m	Var/entier	Le milieu de l'intervalle de recherche

☒ **Algorithme de la fonction rech_dich:**

0) DEF FN rech_dich(T:tab ;n,v:entier): Booléen

1) [a←1, b←n, trouve←faux]

répéter

m← (a+b) DIV 2

si T[m]=v **alors**

trouve←vrai

sinon si T[m]>v **alors**

b←m-1

sinon

a← m+1

finsi

jusqu'a (trouve=vrai) OU (a>b)

2) rech_dich←trouve

3) Fin rech_dich

☒ **En Pascal:**

```
function rech_dich(T:tab;n,v:integer):boolean;
var a,b,m:integer; trouve:boolean;
begin
        a:= 1 ; b:= n; trouve:=false;
        repeat
                m:=(a+b) div 2;
                if T[m]=v then
                        trouve:=true
                else if T[m]>v then
                        b:=m-1
                else
                        a:=m+1;
        until (trouve=true) or (a>b);
rech_dich:=trouve;
end;
```