DEVOIR : TD2

Ex1 :

```
#include<iostream>
using namespace std;
int compteur;
void appelfonction(){
        ++compteur;

}
int main(){
    appelfonction();
    appelfonction();
    appelfonction();
    cout<<"c'est l'appel numero"<<compteur<<endl;
    return 0;
}
```

EX2 :

```
#include<iostream>
using namespace std;
bool verifierf1(int x){
    if(x%2==0){
       return 1;}
       else return 0;

    }
bool verifierf2(int x){
    if(x%3==0){
        return 1 ;
    }
        else
        return 0;
}
int main (){
    int x;
    while(x>0){
    cout<<"saisis x"<<endl;
    cin>>x;

    if((verifierf1 (x))&& (verifierf2(x))){
        cout<<"le nombre est pair ,multiple de 3 et divisible par 6"<<endl;
                }
        else if (verifierf1(x)){
               cout<<"x est pair"<<endl;
        }
```

```
        else if(verifierf2(x)){
            cout<<"x est multiple 3"<<endl;
        }

    else {
    cout<<"x ni multiple de 3 ni pair"<<endl;
    }}
    return 0;
}
```

EX3 :

En utilisant le « formalisme tableau » :

```
#include<iostream>
using namespace std;
int main(){
    int i;
    int t[10];
    int pluspetit = t[0];
    int plusgrand =t[0];
for(i=0;i<10;i++){
    cout<<"veuillez remplir:"<<endl;
    cin>>t[i];
    cout<<"t["<<i<<"]= "<<t[i]<<endl;
    }

    for(i=0;i<10;i++){
        if((t[i])<pluspetit){
          pluspetit=t[i];}
          if (plusgrand<t[i])
          {plusgrand=t[i];}
        }
      cout<<"le plust grand element est"<<plusgrand<<endl;
      cout<<"le plus petit element est"<<pluspetit<<endl;
        return 0;

}
```

EN utilisant le « formalisme pointeur » :

```
#include<iostream>
using namespace std;
int main(){
    int i;
    int t[10];
```

```
     int *p=t;
    int pluspetit = *p;
    int plusgrand =*p;
for(i=0;i<10;i++){
    cout<<"veuillez remplir:"<<endl;
    cin>>*p;
    cout<<"t["<<i<<"]= "<<*p<<endl;
    }
   p=t;
    for(i=0;i<10;i++){
        if(*p<pluspetit){
          pluspetit=*p;}
          if (plusgrand<*p)
          {plusgrand=*p;}

        }
      cout<<"le plust grand element est"<<plusgrand<<endl;
      cout<<"le plus petit element est"<<pluspetit<<endl;
        return 0;

}
```

// le code ça marche mais pour le plus petit et plus grand élément il m'affiche des valeurs aléatoires et pas parmis les valeurs que j'avais déjà saisis je n'arrive pas à comprendre pourquoi//

EX4 :

```
#include <iostream>
using namespace std;
int main(){
int n,i;
int *t;
cout<<"veuillez saisir la taille:"<<endl;
cin>>n;
t=new int[n];
for(i=0;i<n;i++){
    cout<<"veuillez saisir t[i] :"<<endl;
    cin>>t[i];
}
for(i=0;i<n;i++){
    cout<<"t["<<i<<"]= "<<t[i]<<endl;
}
//question 2://
```

```
int *t2=new int[n];
for(i=0;i<n;i++){
    t2[i]=t[i]*t[i];
}
//question 3//
delete[]t;
cout<<"les nombres du second tableau :"<<endl;
for(i=0;i<n;i++){
   cout<<"t2["<<i<<"]="<<t2[i]<<endl;
}

delete []t2;
return 0;

}
```

EX5 :

```
#include <iostream>
using namespace std;
int main() {
    int a=2;
    int &ref_a=a;
    int *p_a =&a;

    cout << "Valeur de 'a' = " << a << endl;
    cout << "Adresse de 'a' : " << &a << endl;

    cout << "Valeur de 'ref_a' = " << ref_a << endl;
    cout << "Adresse de 'ref_a' : " << &ref_a << endl;

    cout << "Valeur pointée par 'p_a' = " << *p_a << endl;
    cout << "Adresse de 'p_a' : " << p_a << endl;

    return 0;
}
```

EX6 :

En transmettant l’adresse des variables concernées :

```
#include <iostream>
using namespace std;
void incrementer(int *variable) {
   ++(*variable);
}

void permuter(int *x, int *y) {
```

```
    int c;
    c = *x;
    *x = *y;
   *y = c;
}

int main() {
    int a = 5;
    int b= 2;
    int valeur=6;
incrementer(&valeur);
permuter(&a,&b);
cout<<a<<" "<<b<<endl;
cout<<valeur<<endl;
    return 0;
}
```

En utilisant la transmission par référence :

```
#include <iostream>
using namespace std;
void incrementer(int &variable) {
   ++ (variable);
}

void permuter(int &x, int &y) {
    int c;
    c = x;
    x = y;
   y = c;
}

int main() {
    int a = 5;
    int b= 2;
   int valeur=7;
incrementer(a);
permuter(a,b);
cout<<a<<" "<<b<<endl;
cout<<valeur;
    return 0;
}
```

EX7 :

```cpp
#include<iostream>
Using namespace std;
using namespace std;
int main(){
    bool echange;
    int t[10];
    int i;
    cout<<"SAISI"<<endl;
    for (i=0;i<10;i++)
    {

            cin>>t[i];
    }
    do
    {
        echange=false;

     for(i=0;i<9;i++)
     {
            if(t[i]>t[i+1])
        {
            int t1=t[i];
            t[i]=t[i+1];
            t[i+1]=t1;
            echange=true;
        }
    }
    }
    while(echange);

    for(i=0;i<10;i++)
    {
        cout<<"LE TABLEAU EN ORDRE CROISSANT"<<t[i]<<endl;
    }
    return 0;
}
```

EX 8:

```cpp
#include <iostream>
using namespace std;
class Complexe {
private:
    double reel;
    double imaginaire;

public:
    Complexe(double r, double i) : reel(r), imaginaire(i) {}
```

```
    Complexe addition(const Complexe& autre) const {
        double reel_resultat = reel + autre.reel;
        double imaginaire_resultat = imaginaire + autre.imaginaire;
        return Complexe(reel_resultat, imaginaire_resultat);
    }

    Complexe soustraction(const Complexe& autre) const {
        double reel_resultat = reel - autre.reel;
        double imaginaire_resultat = imaginaire - autre.imaginaire;
        return Complexe(reel_resultat, imaginaire_resultat);
    }

    Complexe multiplication(const Complexe& autre) const {
        double reel_resultat = reel * autre.reel - imaginaire *
autre.imaginaire;
        double imaginaire_resultat = reel * autre.imaginaire + imaginaire *
autre.reel;
        return Complexe(reel_resultat, imaginaire_resultat);
    }

    void afficher() const {
        cout << "réelle : " << reel << ",  imaginaire : " << imaginaire << "i"
<< endl;
    }
};

int main() {
    double reel1, imaginaire1, reel2, imaginaire2;
    char choix;

    cout << "Entrez la partie réelle du nombre 1 complexe : ";
   cin >> reel1;
    cout << "Entrez la partie imaginaire du nombre 1 complexe : ";
    cin >> imaginaire1;
    cout << "Entrez la partie réelle du nombre 2 complexe : ";
   cin >> reel2;
    cout << "Entrez la partie imaginaire du nombre 2 complexe : ";
   cin >> imaginaire2;

    Complexe nombre1(reel1, imaginaire1);
    Complexe nombre2(reel2, imaginaire2);
    Complexe resultat(0, 0);

   cout << "Choisissez une opération :"
             << " Addition"
             << "Soustraction"
             << "Multiplication"
             << endl;
   cin >> choix;
```

```
    switch (choix) {
        case 1:
            resultat = nombre1.addition(nombre2);
            cout << "Résultat de l'addition : ";
            break;
        case 2:
            resultat = nombre1.soustraction(nombre2);
            cout << "Résultat de la soustraction : ";
            break;
        case 3:
            resultat = nombre1.multiplication(nombre2);
            cout << "Résultat de la multiplication : ";
            break;
        default:
            cout << "choix invalide" << endl;
            return 1;
    }

    resultat.afficher();

    return 0;
}
```

EX12:

```
#include<iostream>

using namespace std;
static int compteur=0;
class appel{

   public: appel()
  {
    ++compteur;
  }
};

int main()
{
  appel F;

cout<<"le compteur = "<<compteur<<endl;
return 0;

}
```

EX13:

```
#ifndef POINT_H
```

```
#define POINT_H

class Point {
public:

    Point(float x = 0.0, float y = 0.0);


    void deplace(float a, float b);

    // Fonction pour afficher les coordonnées du point
    void affiche() const;

private:
    float x;
    float y;   };

#endif

#include "Point.h"
#include <iostream>

Point::Point(float x, float y) : x(x), y(y) {}


void Point::deplace(float a, float b) {
    x += a;
    y +=b;
}

void Point::affiche() const {
    std::cout << "Point (" << x << ", " << y << ")" << std::endl;
}


#include "Point.h"

#include<iostream>
int main() {
    // Déclaration d'un point avec des coordonnées par défaut (0, 0)
    Point point1;

    // Affichage du point
    point1.affiche();

    // Déplacement du point
    point1.deplace(2.5, 3.5);

    // Affichage du point après le déplacement
```

```
    point1.affiche();

    return 0;
}
```

```
    return 0;
}
```