CC

4

Devoir AP1

Devoir de 45 minutes - barème donné à titre indicatif

4.1 Exercice 1 : lecture de code C++

Lisez attentivement les 6 extraits de code ci-dessous. Tous ces codes sont INDEPENDANTS les uns des autres.

➤ Code 1 :

```
1  int a;
2  int *b=new int[3];
3  b=&a;
4  delete b;
```

➤ Code 2 :

```
1  int **b=new int[3];
2  b[0]=new int;
3  b[1]=new int;
4  b[2]=new int;
5  delete b[0];
6  delete b[1];
7  delete b[2];
8  delete []b;
```

➤ Code 3 :

```
1  void fct(int *a)
2  {
3    a=new int;
4    *a=0;
5    delete a;
6  }
7  int main()
8  {
9    int a=1;
10   fct(&a);
11   cout << a;
12   return 0;
13 }
```

➤ Code 4 :

```
1  void fct(int *a)
2  {
3    *a=0;
4  }
5  int main(){
6    int *a=new int;
7    fct(&a);
8    return 0;
9  }
```

➤ Code 5 :

```
1  void fct(int *a){
2    a=new int;
3    *a=0;
4  }
5  int main(){
6    int *a;
7    fct(a);
8    delete a;
9    return 0;
10 }
```

➤ Code 6 :

```
1  int **a=new int*[2];
2  for (int i=0;i<2;i++)
3    a[i]=new int;
4  delete []a;
```

Question 1 */ 2 point(s)*
Indiquer, pour chacun des 6 codes, si ce code compile ou pas (et expliquer pourquoi il ne compile pas le cas échéant)

Question 2 */ 3 point(s)*
Indiquer, pour chacun des 6 codes, si ce code génère ou non une fuite mémoire ou une erreur mémoire (et expliquer l'origine de l'erreur le cas échéant)

Etudiez le code suivant :

```
1  #include <cstdlib>
2  #include <iostream>
3  using namespace std;
4
5  void fct1(int *p, int taillep)
6  {
7    for (int i=0;i<taillep;i++)
8      cout << *(p+i) << "|";
9    cout << endl;
10 }
11
12 void fct2(int *&p, int & taillep, int *t, int taillet)
13 {
14   int *f=new int[taillep];
15   for (int i=0; i< taillep ; i++)
16     f[i]=p[i];
17   delete [] p;
18   p=new int [taillep+taillet];
19   for (int i=0; i< taillep ; i++)
20     p[i]=f[i];
21   for (int i=taillep; i< taillep+taillet; i++)
22     p[i]=t[i-taillep];
23   taillep+=taillet;
24   delete [] f;
25
26 }
27
28 int main(){
29
30   int tailleTab1=3;
```

```
31    int *tab1=new int[tailleTab1];
32    tab1[0]=0;tab1[1]=1;tab1[2]=2;
33    int tailleTab2=2;
34    int *tab2=new int[tailleTab2];
35    tab2[0]=3;tab1[1]=4;
36    fct1(tab1,tailleTab1);
37    fct1(tab2,tailleTab2);
38    fct2(tab1,tailleTab1,tab2,tailleTab2);
39    fct1(tab1,tailleTab1);
40    delete []tab1;
41    delete []tab2;
42    return 0;
43  }
```

Question 3 */ 3 point(s)*
Indiquez ce que ce programme affiche à l'écran.

Question 4 */ 1 point(s)*
Expliquez en une phrase ce que fait ce programme

Question 5 */ 1 point(s)*
Expliquez CLAIREMENT pourquoi il est indispensable, dans la fonction `fct2`, de passer le pointeur p en référence.

4.2 Exercice 2 : lecture de code c++

Soit le code suivant :

```
1   #include <cstdlib>
2   #include <iostream>
3   using namespace std;
4
5   bool mystere(int s[], int taille, int deb, int fin){
6     if (taille < 2)
7       return true;
8     else
9       if (s[deb] == s[fin])
10        return mystere(s,taille- 2,deb+1,fin-1);
11      else
12        return false;
13  }
14
15
16  int main(){
17    const int TAILLE=6;
18    int tab1[TAILLE]={0,1,3,3,1,0};
19    cout << mystere(tab1,TAILLE,0,TAILLE-1) << endl;
20    return 0;
21  }
```

Question 6 */ **5 point(s)***
Effectuez (de manière lisible !) la trace du code proposé ci-dessus. Indiquez également ce qui s'affichera à l'écran.
Recopiez sur votre copie le tableau ci-dessous (surtout ne répondez pas sur l'énoncé !)

	S	taille	deb	fin
Appel 1 de mystère				
Appel 2 de mystère				
Appel 3 de mystère				
Appel ... de mystère				

Question 7 */ **1 point(s)***
Si à la fin du main on rajoute le code suivant, que va-t-il s'afficher à l'écran ? (pas besoin d'écrire la trace)

```
1 int tab2[TAILLE]={0,2,3,3,1,0};
2 cout << mystere(tab2,TAILLE,0,TAILLE-1) << endl;
```

Question 8 */ **1 point(s)***
Expliquez en une phrase ce que teste ce programme

Question 9 */ **3 point(s)***
Proposez une modification de la fonction `mystere` et du `main` afin que s'affiche le nombre d'appels récursifs réalisés.
Par exemple,

- si `tab1` vaut {0,1,3,2,2,3,1,0} alors dans la fonction `main` un message devra permettre d'afficher : "il y a eu 4 appel(s) récursif(s)".

- si `tab1` vaut {0,2,3,3,1,0} alors dans la fonction `main` un message devra permettre d'afficher : "il y a eu 1 appel(s) récursif(s)".