TP : Graphe de flot de contrôle.

Exercice 1. On donne le code suivant :

```
int i = 1, s = 0;
while (i <= 1000){
  s=s+i;
  i=i+1;
}
printf("%d",s);
```

1. Etablir son graphe flot de contrôle (GFC).
2. Donner l'expression algébrique des chemins de contrôle possibles.

Exercice 2. On donne le programme

```
void f(int c, int x){
 if (c==1)
    x=x+1;
 if (c==2)
    x=x-1;
 return x;
   }

```

1. Donner le graphe de flot de contrôle
2. Donner l'expression des chemins de contrôle. En déduire leur nombre.
3. Donner les chemins de contrôle non faisables. Conclure.
4. Donner une nouvelle version de ce programme de façon que tous les chemins de contrôle soient faisables.

Exercice 3.

1. Donner un graphe de contrôle G et des données de test DT montrant que le critère de couverture tous-les-nœuds (puis tous-les-arcs) est insuffisant pour détecter une erreur pour le code suivant :

```
float f(float x){
  if (x>=0){
    x = pow(-1.0,(int) x)*x;
    return sqrt(x);
  }
  else return sqrt(-x);
    }

```

2. Calculer ensuite les taux de couverture tous-les-arcs puis tous-les-nœuds de ce graphe.

3. En tenant compte du fait que $(-1)^n$ prend 2 valeurs, proposer un nouveau graphe de flot de contrôle pour ce code et détecter le problème par une couverture tous-les-arcs.

Exercice 4. On considère la fonction

```
float f(int inf, int sup, int tab []){
  int i = inf;
  float som = 0;
  while (i<=sup){
    som = som+tab[i];
    i++;
  }
  return 1/som;
    }

```

1. Donner son GFC.
2. Quel chemin sensibilise DT1=$\{\text{inf} = 0; \text{sup} = 2; tab = \{1; 2; 3\}\}$? Est-ce que ce chemin satisfait les critères tous les nœuds ? Tous les arcs ?
3. Quel défaut n'est pas détecté ? Quel critère de couverture vu en cours permettrait de le détecter ?

Exercice 5. On considère l'extrait de code suivant :

```
if ( (A||B) && C){
  // instructions
}
else {
  // instructions
}
```

1. On dit que la *couverture des conditions (atomiques)* est vérifiée si chaque variable est évaluée au moins une fois à **true** et une fois à **false**.
   Proposer un nombre minimum de tests vérifiant la couverture des conditions.
2. On dit que la *couverture des décisions* est satisfaite, lorsque la proposition `( (A||B) && C)` est évaluée une fois à **true** et une fois à **false**.
   Proposer un jeu de test pour une couverture des décisions.
3. Une *couverture des conditions/décisions modifiées* (MC/DC pour « Modified condition/decision coverage ») impose que chaque variable booléenne soit évaluée une fois à **true** et une fois à **false** tout en ayant une influence sur le résultat final.
   D'un cas de test à l'autre, le changement d'une seule condition atomique change aussi la décision : $n-1$ variables étant fixées, il faut que la n-ème influe sur le résultat.
   Pour n opérandes bouléens, il faut donc trouver au minimum $n+1$ tests pour assurer la couverture MC/DC alors qu'une couverture intégrale impose 2^n tests.
   Donner un jeu de test MC/DC pour notre exemple.

4. On considère la condition **((u == 0) OU (x>5)) ET ((y<6) OU (z == 0))** .
   Combien de test faut-il pour une couverture intégrale ?
   Donner une couverture MC/DC.

Exercice 6. Écrire la fonction `binary_search(int v, int [a], int n)` qui cherche par dichotomie l'élément v dans le tableau (supposé trié par ordre croissant a). La fonction renvoie -1 si v est absent du dans le tableau et son indice dans le tableau sinon.

Écrire un jeu de tests pour cette fonction en utilisant ce qui a été dit en cours pour les tests aux limites.

Exercice 7. Le programme suivant doit retourner le plus petit élément du tableau trié `t` qui est strictement supérieur à `x`.

```
1 let f t x =
2   (*hyp : t trié par ordre croissant*)
3   let i = ref 0 and n = Array.length t in
4   while !i <= (n-1) && t.(!i)<=x do
5     incr i;
6   done;
7   t.(!i);;
```

1. Etablir son graphe de contrôle.
2. Faire une couverture tous-les-sommets.
3. Est-il utile de faire une couverture tous les arcs ?
4. Le *critère de couverture des limites* (Boundary Coverage) est une stratégie de test visant à détecter les erreurs liées aux conditions aux limites d'un programme. Il se concentre sur les valeurs frontières des domaines de définition des variables, car les bugs se manifestent souvent à ces endroits.
   Ce critère peut-être utilisé indépendamment de la recherche d'une couverture de chemins.
   Règles générales du critère des limites :
   — Pour toute variable entière `i` dans une condition `a <= i <= b` :
   Tester `i = a-1, i = a, i = b, i = b+1` .
   — Pour les tableaux :
   Tester les cas `taille = 0, taille = 1` , et `taille > 1` .

   (a) Identifier les critères de couvertures aux limites.

   (b) Trouver les deux problèmes de ce code.