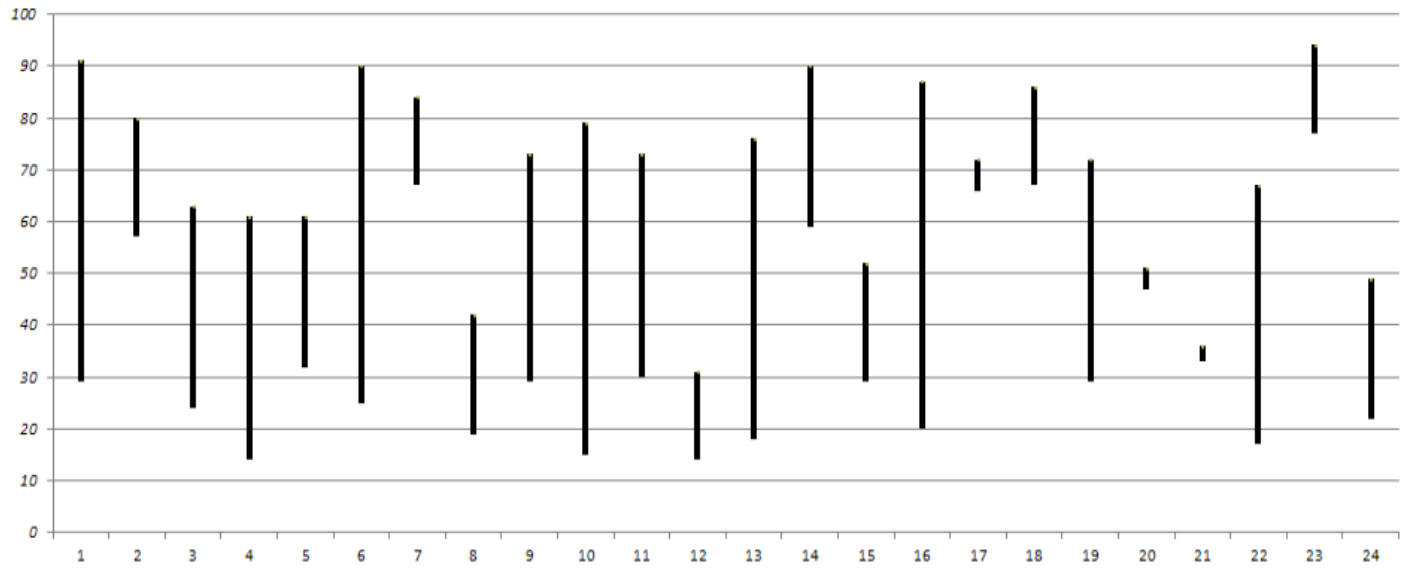
Algorithmique et Programmation – Examen (durée 3 heures)

CORRECTION de la deuxième Session du 24 Juin 2015

Remarque : Ce document ne présente que des éléments de correction. Les variantes possibles sont nombreuses.

Exercice 1 : Un programme qui utilise des briques de base

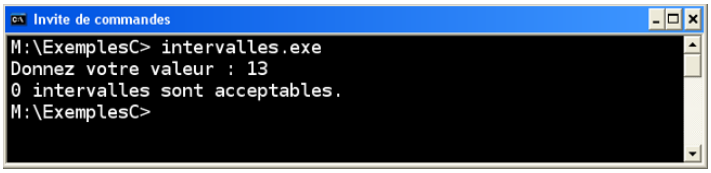
Pour résoudre un problème (mécanique, électronique, gestion industrielle, ...), des intervalles d'acceptabilités ont été définis. Ces intervalles, au nombre de 24, sont chacun défini par une borne min et une borne max. Ces intervalles ont été représentés graphiquement sur la figure suivante :

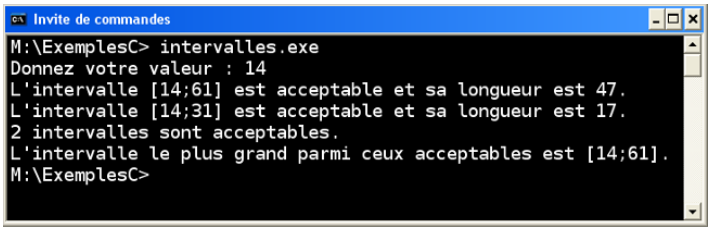


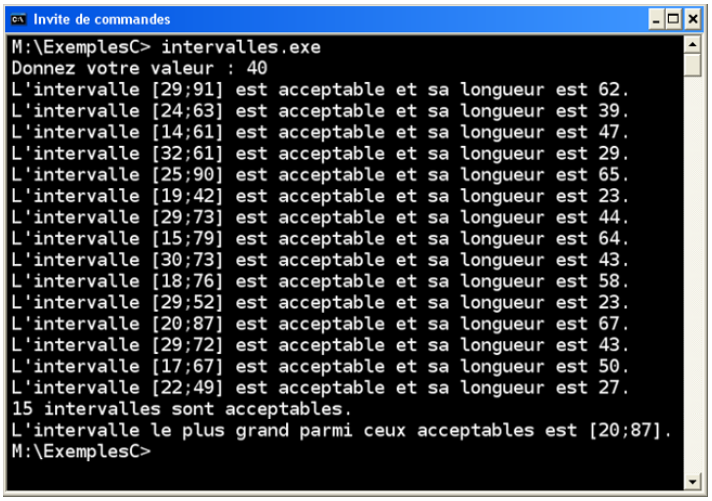
Écrire **UN PROGRAMME** dans lequel sont déclarés les deux tableaux suivants, définissant les 24 bornes min et les 24 bornes max :

```
unsigned int min[24] = { 29, 57, 24, 14, 32, 25, 67, 19, 29, 15, 30, 14, 18, 59, 29, 20,
                         66, 67, 29, 47, 33, 17, 77, 22 } ;
unsigned int max[24] = { 91, 80, 63, 61, 61, 90, 84, 42, 73, 79, 73, 31, 76, 90, 52, 87,
                         72, 86, 72, 51, 36, 67, 94, 49 } ;
```

et qui demande à l'utilisateur de saisir une valeur, puis affiche à l'aide d'une boucle les intervalles dans lesquels la valeur est acceptée (càd comprise entre la borne min et la borne max). À l'aide de cette même boucle, le programme comptera le nombre d'intervalles acceptés et enregistrera l'indice de celui qui est le plus long. Avant de s'arrêter, le programme affichera le nombre d'intervalles acceptés et affichera celui qui est le plus long. Ci-après trois exemples d'exécution de ce programme lorsque l'utilisateur entre les valeurs 13, 14 et 40 :








```c
#include <stdio.h>

int main () {
  int i, cpt=0, indice_max ;
  unsigned int min[24] = { 29, 57, 24, 14, 32, 25, 67, 19, 29, 15, 30, 14, 18, 59, 29, 20, 66, 67,
      29, 47, 33, 17, 77, 22 } ;
  unsigned int max[24] = { 91, 80, 63, 61, 61, 90, 84, 42, 73, 79, 73, 31, 76, 90, 52, 87, 72, 86,
      72, 51, 36, 67, 94, 49 } ;
  unsigned int val, longueur, longueur_max=0 ;

  printf ("Donnez votre valeur : ") ;
  scanf ("%u", &val) ;

  for (i=0 ; i<24 ; i++) {

    if (val >= min[i] && val <= max[i]) {

      longueur = max[i] - min[i] ;

      if (longueur > longueur_max) {
        longueur_max = longueur ;
        indice_max = i ;
      }

      printf ("L'intervalle [%u;%u] est acceptable et sa longueur est %d.\n", min[i], max[i],
          longueur) ;

      cpt++ ;
    }

  }

  printf ("%d intervalles sont acceptables.\n", cpt) ;

  if (cpt > 0) {
    printf ("L'intervalle le plus grand parmi ceux acceptables est [%u;%u].\n", min[indice_max], max[
        indice_max]) ;
  }

  return 0 ;
}
```

E	**Être capable d'écrire un programme simple**

Exercice 2 : Des fonctions qui manipulent un annuaire

Soit la structure de donnée suivante, permettant de décrire une personne par son nom et son année de naissance :

```
typedef struct Personne {
  char nom [256] ;
  unsigned int annee ;
} *TPersonne ;
```

Écrire **UNE FONCTION** qui affiche l'annuaire passé en paramètre sous forme d'un tableau de TPersonne de taille N. Le prototype de la fonction sera le suivant :

```
void afficher (TPersonne *annuaire, int N) ;
```

```
50 void afficher (TPersonne *annuaire, int N) {
51   int i ;
52
53   for (i=0 ; i < N ; i++) {
54     printf ("%s␣(%i)\n", annuaire[i]->nom, annuaire[i]->annee)  ;
55   }
56
57   printf ("\n") ;
58 }
```

Écrire **UNE FONCTION** qui trie l'annuaire passé en paramètre sous forme d'un tableau de TPersonne de taille N, par ordre croissant de noms. En cas de noms identiques, c'est alors par ordre croissant d'année que le tri est effectué. Le prototype de la fonction sera le suivant :

```
void trier (TPersonne *annuaire, int N) ;
```

```
13 void trier (TPersonne *annuaire, int N) {
14   int i, j ;
15   TPersonne tmp ;
16
17   for (i=0 ; i < N ; i++) {
18     for (j=0 ; j < N-i-1 ; j++) {
19       if (strcmp (annuaire[j]->nom, annuaire[j+1]->nom) > 0) {
20         tmp = annuaire[j] ;
21         annuaire[j] = annuaire[j+1] ;
22         annuaire[j+1] = tmp ;
23       }
24       else if (!strcmp (annuaire[j]->nom, annuaire[j+1]->nom)) {
25         if (annuaire[j]->annee > annuaire[j+1]->annee) {
26           tmp = annuaire[j] ;
27           annuaire[j] = annuaire[j+1] ;
28           annuaire[j+1] = tmp ;
29         }
30       }
31     }
32   }
33 }
```

C/D **Être capable de manipuler une structure de données simple et d'écrire un algorithme**