Réalisé par :

Oumaïma Bounhar – 2ia - SD

TP C Progm: Organisation de journées culturelles

Exercice 1 :

Description du programme :

Les fonctions :

On commence par créer une fonction qui va ouvrir tous les fichiers. Celle-ci est appelée **open_file()** et prend en entrée le chemin du fichier.

Ensuite la fonction **getChoice()** prend en argument une ligne et une colonne du fichier et retourne la valeur correspondante. Cette fonction servira pour la fonction suivante.

Puis on crée la fonction **getRows()** qui prend en entrée le nombre de lignes du fichier, le nombre de ses colonnes et le chemin du fichier. Cette fonction renvoie la copie du fichier en matrice en éliminant l'en-tête.

Et finalement, on déclare la fonction **fonction_objectif()** qui prend en entrée le coût et la pénalité du jour et retourne la fonction objectif. Celle-ci sera précisé plus tard dans le main.

Le main :

On ouvre le fichier, on compte le nombre de ligne qu'on stocke dans la variable num_lines puis on appelle la fonction getRows() qu'on stocke dans la matrice families_pref. Cette dernière contient alors toutes les familles et leurs choix respectifs. C'est la transformation du fichier csv en format de matrice pour manipuler plus facilement les données.

On alloue de la mémoire pour la matrice chosen_Days qui contiendra les familles en lignes et les jours en colonnes suite au tri par ordre de préférence.

```
138     for(int i = 0 ; i<num_lines ; i++){ //We fix a line of a family
139         for(int j = 1 ; j<6 ; j++){ //We fix a column of a preferency
140             for(int k = 0 ; k<7 ; k++){ //we search for the chosen day according to a preferency
141                 if(families_pref[i][j] == k && count[k]<300){
142                     chosen_Days[index_member[k]][k] = families_pref[i][0];
143                     index_member[k]+=1;
144                     count[k] += families_pref[i][0];
145                     printf("-----------------------------------------------------------\n");
146                     switch(j) //According to the value of j which correspond to a preferency, we calculate the cost
147                     {
148                     case 1 :
149                         break;
150                     case 2 :
151                         cost+= 50;
152                         break;
153                     case 3 :
154                         cost+= 50 +9*families_pref[i][0];
155                         break;
156                     case 4 :
157                         cost += 100 + 9*families_pref[i][0];
158                         break;
159                     case 5 :
160                         cost += 200 +9*families_pref[i][0];
161                         break;
162                     }
163                 }
164             }
165         }
166     }
```

L'objectif de cette partie est d'attribuer les familles aux jours en privilégiant leurs premiers choix.

On fixe la ligne i de la matrice families_pref, ensuite on fixe la colonne j, par ordre croissant. En faisant varier k de 0 à 6 qui correspond aux jours, on cherche dans families_pref si la famille a choisi le k-ème jour et si ce jour est incomplet (ie que le nombre de personne est inférieur à 300). Si c'est bien le cas, on rajoute cette famille à la colonne du jour correspondant puis on incrémente la liste index_member[k] afin de l'utiliser pour sauter une ligne dans chosen_Days. Aussi, on utilise la liste count[] pour compter le nombre de personne dans chaque jour k. Puis suivant la colonne j à laquelle on est arrivée, on calcule le coût grâce au switch case. Si la famille n'a pas choisi le jour k et la condition count[k]<300 n'est pas remplie aussi, alors on sort de la boucle de k puis on regarde la colonne j suivante.

A la suite de cette partie, on affiche la matrice et on calcule la fonction objectif grâce à la valeur de cost, le tableau penalty et le tableau day.

Finalement, on libère l'espace mémoire alloué aux deux matrices families_pref et chosenDays et ferme le fichier.

Résultats lors de l'exécution :

On affiche les valeurs de la matrice families_pref grâce à la fonction getRows(), (printf value) afin de s'assurer qu'on a bien les bonnes valeurs de tout le fichier. C'est bien le cas ici.

Ensuite, on affiche la matrice chosenDays qui est contient les membres de chaque famille dans le meilleur jour choisi par ordre croissant de préférences.

Fonction objectif = 60013 pour le fichier pb20.csv par exemple.

Exercice 2 :

Une des solutions est de trier par ordre décroissant les pourcentages pour chaque événement et retourner dans cette ordre les jours associés. Ainsi, avec un algorithme de tri simple en comparant les éléments deux à deux (ou à la main), on obtient l'ordre suivant :

Événement 1	5	2	1	6	0	3	4
Événement 2	3	4	5	2	1	0	6
Événement 3	6	5	4	3	0	2	1