```java
import java.util.Arrays;               // 11/10/2023 - rene.natowicz@esiee.fr
/* Juliette dispose de H heures pour réviser n unités.
Pour toute unité i, 0 ≤ i < n et tout nombre h d'heures de révision, 0 ≤ h < H+1,
Juliette a estimé la note e(i,h) qu'elle obtiendra au contrôle de l'unité i si elle
consacre h heures à la révision de cette unité.
Ces estimations sont dans un tableau E[0:n][0:H+1] de terme général E[i][h] = e(i,h).
Les coefficients des unités sont dans le tableau C[0:n].
Les données du problème sont les estimations E et les coefficients.
Le nombre d'heures H est E[0].length - 1.

Supposons le problème résolu.
------------------------------
Notation m(n,H) : somme pondérée maximum des notes aux contrôles des n premières
unités, pour H heures de révision réparties sur les n premières unités.

(Pour avoir la moyenne, il faudra diviser m(n,H) par la somme des coefficients.)

m(n,H) = Max { c_{n-1} * e(n-1, h) + m(n-1, H-h)  }
         sur h, 0 ≤ h < H+1.

Généralisation : m(k,t) est la somme pondérée maximum des notes aux contrôles
des k premières unités, pour t heures de révision réparties  sur les k premières unités.

Equation de récurrence :
    Base k = 0, 0 ≤ t < H+1 :
        m(0,t) = 0 car le sous-ensemble [0:0=k] d'unités est vide.
    Hérédité, 1 ≤ k < n+1, 0 ≤ t < H+1 :
          m(k,t) = max { c_{k-1} * e(k-1,h) + m(k-1,t-h) }
          sur h, 0 ≤ h < t+1

Explication de l'hérédité : h heures accordées à la k-ème unité "rapportent"
c_{k-1} * e(k-1,h) ; il reste t-h heures à répartir au mieux sur les k-1 premières
unités. Cette répartition de t-h heures sur le sous-ensemble [0:k-1] des k-1 premières
unités a la valeur m(k-1, t-h).

Voir exemple d'exécution en fin de fichier.
*/
class Juliette{
    public static void main(String[] Args){
        int[] C = {4, 3, 4}; // coefficients des unités
        int[][] E = {
            {8, 10, 10, 12, 12, 12, 12, 12, 12, 16, 16}, // unité 0
            {16, 16, 18, 20, 20, 20, 20, 20, 20, 20, 20}, // unité 1
            {8, 12, 14, 14, 16, 18, 18, 18, 18, 20, 20} // unité 2
        };
        int n = E.length, H = E[0].length-1;
        System.out.printf("\n%d unités, %d heures de révision\n", n, H);
        System.out.println("Coefficients des unités : " + Arrays.toString(C));
        System.out.println("Tableau E des notes estimées :");
        afficherTableauE(E);
        int[][][] MA = calculerMA(E,C);
        int[][] M = MA[0], A = MA[1];
        System.out.println("\nTableau M de terme général m(k,t) :");
        afficherTableauM(M);
        System.out.println("\nTableau A de terme général a(k,t) = arg m(k,t) :");
        afficherTableauA(A);
        System.out.printf("\nValeur maximum m(%d,%d) = %d\n", n, H, M[n][H]);
        int SC = somme(C); // somme des coefficients
        System.out.printf("Moyenne : %d/%d = %.2f\n", M[n][H], SC, (float)M[n][H]/(float)SC);
        System.out.println("\nUne répartition optimale du temps de révision :");
        aro(A,E,C,n,H);
        System.out.println();
//      aro(A,E,C,n,6); // Répartition optimale de 6 heures de révision...
//      System.out.println();
    }

    static int[][][] calculerMA(int[][] E, int[] C){
    // calcule et retourne les deux tableaux M et A de terme généraux m(k,t) et arg m(k,t)
        int n = E.length, H = E[0].length - 1;
        int[][] M = new int[n+1][H+1], A = new int[n+1][H+1];
        // M de terme général M[k][t] = m(k,t) et A de terme général A[k][t] = arg m(k,t).
        // Base de la récurrence : m(0,t) = 0, qqsoit t, 0 ≤ t < H+1
        for (int t = 0; t < H+1; t++) {M[0][t] = 0;
            A[0][t] = 0;} // les valeurs A[0][t] sont quelconques
        for (int k = 1; k < n+1; k++) // cas général, résolution par k croissant
            for (int t = 0; t < H+1; t++){ // pour tout nombre d'heures t à répartir
                // ici : k et t sont fixés.
                // calcul de m(k,t) = max_{ 0 ≤ h < t+1}{ c_{k-1} * e(k-1,h) + m(k-1,t-h) }
                M[k][t] = Integer.MIN_VALUE;
                for (int h = 0; h < t+1; h++){
                    /* mkth : somme pondérée maximum si h heures sont allouées à la révision
                    de la k-ème unité dans une répartition optimale de t heures de révision
                    sur les k premières unités */
                    int mkth = C[k-1]*E[k-1][h]+M[k-1][t-h];
                    if (mkth > M[k][t]){
                        M[k][t] = mkth;
                        A[k][t] = h; /* arg m(k,t) : h heures données à la k-ème
                        unité dans la répartition optimale de t heures sur le
                        sous-ensemble des k premières unités. */
                    }
                }
            }
        return new int[][][] {M,A};
    } // Temps de calcul : alpha x n H**2 + ... ( "..." est négligeable pour n et H "grands")

    static void aro(int[][] A, int[][] E, int[] C, int k, int t){
    /* affichage d'une répartition optimale (RO) de t heures sur le sous-ensemble
    des k premières unités.
    Notation : RO(k,t) est une répartition optimale de t heures sur les unités [0:k]. */
        if (k==0) return; // RO(0,t) = ∅. Sans rien faire, RO(0,t) a été affichée.
        int h = A[k][t]; // RO(k,t) = RO(k-1,t-h) union { "k-1<--h" }
        aro(A,E,C,k-1,t-h); // RO(k-1,t-h) a été affichée
        System.out.printf("C[%d] * e(%d,%d) = %d * %d = %d\n",
            k-1, k-1, h, C[k-1], E[k-1][h], C[k-1]*E[k-1][h]); // { "k-1<--h" } affiché
    // RO(k-1,t-h) a été affichée, "k-1<--h" a été affiché, donc RO(k,t) a été affichée
    }
    static void afficherTableauE(int[][] T){ int n = T.length;
        for (int i = n-1; i > -1; i--)
            System.out.printf("unité %d : %s\n", i, Arrays.toString(T[i]));
    }
    static void afficherTableauM(int[][] T){ int n = T.length;
        for (int k = n-1; k > -1; k--)
            System.out.printf("k=%d : %s\n", k, Arrays.toString(T[k]));
    }
    static void afficherTableauA(int[][] T){ int n = T.length;
        for (int k = n-1; k > -1; k--)
            System.out.printf("k=%d : %s\n", k, Arrays.toString(T[k]));
    }
    static int somme(int[] T){int n = T.length;
        int s = 0; for (int i=0; i<n; i++) s=s+T[i];
        return s;
    }
}
```

```
/*

% javac Juliette.java
% java Juliette

3 unités, 10 heures de révision
Coefficients des unités : [4, 3, 4]
Tableau E des notes estimées :
unité 2 : [8, 12, 14, 14, 16, 18, 18, 18, 18, 20, 20]
unité 1 : [16, 16, 18, 20, 20, 20, 20, 20, 20, 20, 20]
unité 0 : [8, 10, 10, 12, 12, 12, 12, 12, 16, 16]

Tableau M de terme général m(k,t) :
k=3 : [112, 128, 136, 144, 144, 152, 160, 160, 168, 172, 174]
k=2 : [80, 88, 88, 96, 100, 102, 108, 108, 108, 112, 112]
k=1 : [32, 40, 40, 48, 48, 48, 48, 48, 48, 64, 64]
k=0 : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Tableau A de terme général a(k,t) = arg m(k,t) :
k=3 : [0, 1, 1, 2, 1, 2, 5, 4, 5, 5, 5]
k=2 : [0, 0, 0, 0, 3, 2, 3, 3, 3, 0, 0]
k=1 : [0, 1, 1, 3, 3, 3, 3, 3, 3, 9, 9]
k=0 : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Valeur maximum m(3,10) = 174
Moyenne : 174/11 = 15,82

Une répartition optimale du temps de révision :
C[0] * e(0,3) = 4 * 12 = 48
C[1] * e(1,2) = 3 * 18 = 54
C[2] * e(2,5) = 4 * 18 = 72

%
*/
```