Compilez et exécutez le code suivant :

```
#include <stdio.h>
/* Declaration des variables */
 unsigned char    u08_var = 0;
 signed char s08_var = 0;
/* Fonction principale */
int main()
{
    printf("Taille des variables (en octets) :\n");
    u08_var = sizeof(char);
    printf("  a. variable de type char  = %d\n", u08_var);
    u08_var = sizeof(short);
    printf("  b. variable de type short = %d\n", u08_var);
    u08_var = sizeof(int);
    printf("  c. variable de type int   = %d\n", u08_var);
 printf("\nResultats des operations :\n");
 u08_var = 27;
 printf("  a. s08_var = %d\n", u08_var);
 s08_var = 27;
 printf("  b. s08_var = %d\n", s08_var);

 u08_var = 511;
 printf("  c. u08_var = %d\n", u08_var);

 s08_var = 511;
 printf("  d. s08_var = %d\n", s08_var);

s08_var = 131;
printf("  e. s08_var = %d\n", s08_var);

s08_var = -132-64;
 printf("  g. s08_var = %d\n", s08_var);
 s08_var = 95 + 50;
 printf(" h. s08_var = %d\n", s08_var);
    return 0; }
```

Figure 2 *Affichage des résultats suite à l'exécution du code*

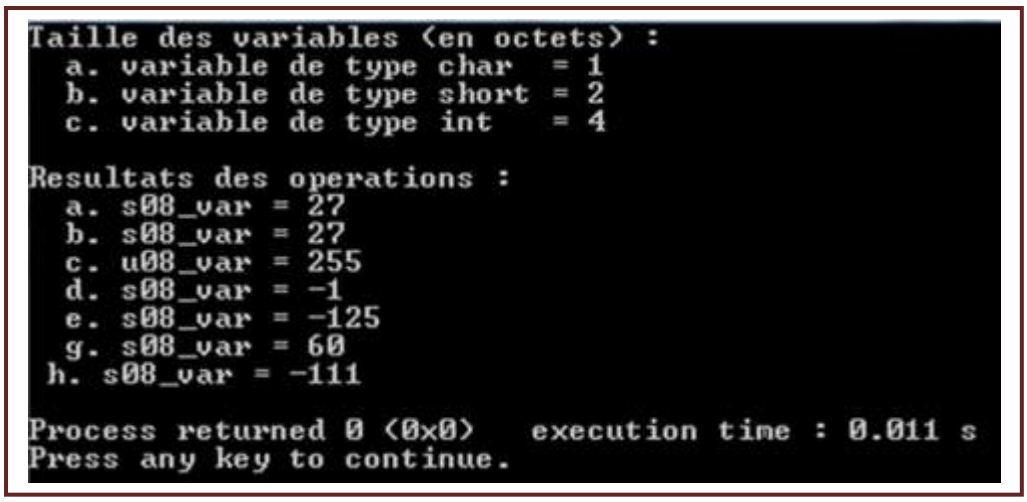

Figure 2 *Affichage des résultats suite à l'exécution du code*

1ère partie : Un problème courant en programmation, le débordement

Dans cette partie, nous nous intéresserons à un problème communément rencontré en programmation lors de la manipulation de variables numériques : le **débordement**. Dans les messages affichés lors de la compilation, le débordement est appelé « `overflow` ».
Nous allons analyser les résultats affichés **(figure 2)** suite à l'exécution du code de la **figure 1**.

1. Donnez les tailles des différents types de variables (char, short et int)
2. Commençons par la 1ère instruction a).
   a. Convertissez la valeur $(511)_{10}$ au format binaire.
   b. Vous remarquerez que $(511)_{10}$ s'écrit sur plus de 8 bits, alors qu'une variable de type « `unsigned char` » ne peut contenir que 8 bits.
      i. Prenez les **8 premiers** bits et reconvertissez-les au format décimal.
      ii. Comparez par rapport au résultat de l'exécution et concluez.
3. Pour les instructions de d et e:
   a. Quel est le résultat que devrait donner, théoriquement, chaque instruction ?
   b. Expliquez les résultats affichés suite à l'exécution du code de la même manière que pour l'exemple précédent.
      i. Rappelez-vous que le type « `char` » est un type signé.
         - Les nombres sont représentés au format **CA2**.
         - Le bit de **poids le plus fort** fait office de **signe**. S'il est à 1, alors le nombre est négatif. Sinon, il est positif.
      ii. Prenez les **8 premiers** bits de chaque résultat et convertissez-les du format CA2 au format binaire.
      iii. Convertissez les valeurs binaires retrouvées au format décimal et comparez-les aux résultats affichés.
5. Pour les instructions g et h, effectuez les soustractions en utilisant la représentation en CA2.
6. Changez le type de variables utilisées. Nous utilisons des variables short au lieu de char. Re-exécutez le code. Que constatez-vous ?