Tom Niget

Partie 3

1.
C'est cohérent, argc=2 car il y a deux arguments : le chemin du programme, et 456.
*Argv pointe vers le premier, le chemin, et **argv pointe vers le premier caractère; or c'est un chemin donc ce caractère est un slash, de code ASCII 47.
2.
prog pointe vers la chaîne du chemin du programme.
arg pointe vers la chaîne "456".
3.
&w désigne l'adresse de la variable w.
*&w désigne la valeur de la variable w.
**&w. désigne la valeur se trouvant à l'adresse spécifiée par la valeur de w.
4.
"bonjour" s'affiche dans le terminal.

Partie 4

5.
Je m'arrête bien.
6.
L'espion est placé. Je vois les 3 premiers éléments de argv, à savoir : chemin, 456, NULL.
7.
J'obtiens bien :

rax	0x7fffffffeee30	140737488285232
rbx	0x80011a0	134222240
rcx	0x80011a0	134222240
rdx	0x7fffffffeec00	140737488284672
rsi	0x7fffffffeebe8	140737488284648

… (beaucoup de lignes)

Partie 5

3. J'observe que mon processeur travaille en little-endian et que les données sont donc stockées avec les bits de poids faible d'abord. Cela se traduit par le fait que l'int32 '12' correspond dans le table d'int8 à '12' suivi de trois '0'.

4. *ci@16 donne 16 éléments sur 8 bits, *si@8 donne 8 éléments sur 16 bits. Même si l'un peut être retrouvé à partir de l'autre, ce sont bien deux choses différentes.

5. L'arithmétique des pointeurs prend en compte la taille du type pointé, donc ii++ incrémente de 4 (sizeof(int) sur ma plateforme) tandis que si++ incrémente de 2 (sizeof(short) sur ma plateforme). Idem pour ci++ qui incrémente de 1 (sizeof(char) sur ma plateforme).

Partie 6

6.1.

8. La boucle va jusqu'à 10 inclus, or en C l'indice du dernier élément d'un tableau vaut N-1. En écrivant 0 dans chaque case, cela écrit donc 0 dans "l'après-dernière" case du tableau, et du fait de la manière dont sont stockées les locales dans les fonctions avec mon compilateur sur ma plateforme (car l'ordre n'est pas défini par la spécification C), cette case virtuelle correspond à la variable sig, qui est donc mise à 0. La condition apparemment impossible de sig != 1234567890 est donc vérifiée.

6.2.

9. C'est un tri de Shell avec suite d'espacements de Knuth.

11. Le problème se trouve ligne 36, mysort est appelé avec argc mais le nombre correct d'éléments est argc-1.

12. Il fallait ajouter :

```
if(head == NULL) {
   head = newNode;
   return;
}
```

Dans InsertAtTail, car sinon cela déréférencait next sur head qui était NULL.

13. Je n'obtiens aucune sortie avec ltrace. Ni depuis la VM Ubuntu, ni depuis WSL, et d'après des camarades je ne suis pas le seul. D'après mes connaissances je peux dire que le programme se sert de la glibc, plus particulièrement de malloc, atoi, printf et free.

14. /etc/ld.so.cache, /lib/x86_64-linux-gnu/libc.so.6

15. À la ligne 15, gau = milieu ***+ 1***.