Assignment#4 Part 1(30%): Put it all together

CS232 Spring 2021

Due: end of February 28

Q1. Warm-up

1.1) Consider the following definitions, where ... is a placeholder for some valid initialization code:

```
char c = ...;
char* p = ...;
char** pp = ...;
char* ppp[100]=...;
```

With respect to the above definitions, what is the type of each of the following expressions? Mark either the appropriate type, or say “ill-typed” if it won't typecheck. The first one has been done as a sample for you.

1. &c: char*
2. c: char
3. *c: ill-typed
4. &p: char**
5. p: char*
6. *p: char
7. &pp: char***
8. *pp: char*
9. *ppp:char*
10. ppp[10]:char*
11. **(ppp): char

1.2) Consider the following code:

```
int x = 7;
int* p1 = &x;
int* p2 = malloc(sizeof(int));
int* p3 = p2;
```

For each of the questions below, answer either “heap” or “stack”. The first one has been done for you.

1. Where is x allocated?: stack
2. Where is p1 **itself** allocated?: stack
3. Where is **what p1 points to** allocated?: stack
4. Where is p2 **itself** allocated?: stack
5. Where is **what p2 points to** allocated?: heap
6. Where is p3 **itself** allocated?: stack
7. Where is **what p3 points to** allocated?: heap

Q2. A memory has the following contents (in little-endian format)

Variable	Address	Bytes	Final Value of Byte
A	0x08000000	**00 00 00 08**	**08 00 00 08**
B	0x08000004	**04 00 00 08**	**04 00 00 08**
C	0x08000008	**fe ff ff ff**	**07 00 00 00**
D	0x0800000C	**ff ff ff ff**	**04 00 00 00**
E	0x08000010	**00 00 00 00**	**18 00 00 08**
F	0x08000014	**01 00 00 00**	**01 00 00 00**
G	0x08000018	**02 03 04 05**	**02 03 04 05**
H	0x0800001C	**33 35 31 00**	**04 00 00 08**

Given the following declarations (assuming a 32-bit architecture):

```
int *A, *B; float C; int D; float E; int F; float G;

struct xform {
    int i[3];
    float * factor1;
    Float * factor2;
    int color;
};

struct xform *H;
```

Fill in columns for the address (in hex) that is changed in each statement and the value (in hex) to which it is changed. **NOTE: The statements are executed in sequence and changes made to memory apply in the following lines.**

C statements	Address(hex)	Value(hex)
`A = B + 1;`	0x08000000	0x08000008
`C = (float) (*A + F);`	0x08000008	0xffffffff
`H = (xform *) &B;`	0x0800001C	0x08000004
`H->factor1 = &E + 2;`	0x08000010	0x08000018
`D = (int) *((char *)(H->factor1)+2);`	0x0800000C	0x00000004
`H->i[(D >> 2)] = D + 3;`	0x08000008	0x00000007