Answers - Work follows

Gilbert Brooks III

Q1

(a) $+47_{10}$

binary: 0010 1111
hex: 0x2F

← 8-bit 2's comp.

(b) -13_{10}

binary: 1111 0011
hex: 0xF3

← 8bit 2's comp.

(c) $+47.0_{10}$ → 32-bit IEEE F.P. rep.

binary: 0100 0010 0011 1100 0000 0000 0000 0000
hex: 0x423C0000

(d) -0.375_{10} → 32-bit IEEE F.P. rep

binary: 1011 1110 1100 0000 0000 0000 0000 0000
hex: 0xBEC00000

(e) "String for 250"

↓ hex ↓

53 74 72 69 6E 67 20 66 6F 72 20 32 35 30 21

(f) 2^{40} is too big for 32-bit signed int

Q2.

(a)
a) a lives in the stack because it's a local var.
b) b_ptr lives on the heap because it's dynamically allocated.
c) *b_ptr lives in heap
d) e_ptr lives in the Data because it is a global var.
e) *e_ptr lives in Data.

(b) Returns 1.

(a) $+47_{10}$ to 8-bit 2s complement integer rep. in binary & hex

binary: ~~0 0 0 0 1 0 1 1 1 1~~ → 00101111

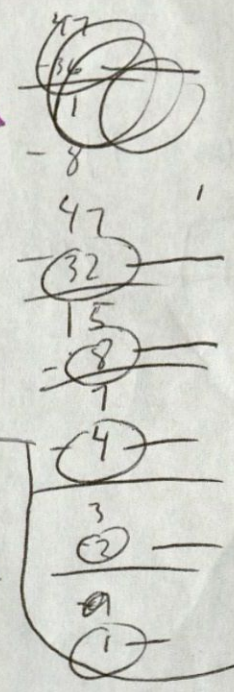
base 128 64 32 16 8 4 2 1

~~0 0 0 1~~ 0010 1111

hexadecimal: 0x 2 F → 0x2F

base 16

Assignment 1 Ans.



(b) -13_{10} to 8-bit 2s complement

binary: ① start w/ positive 13 & work back ~~to 1~~ to -13.

13 → 0000 1101 → 00001101 ⟺ 13

128 64 32 16 8 4 2 1

② lets ~~negate~~ negate it (invert) → 11110010

+1

11110011

① add 1

= 1111 0011 ← -13

hex: 0x F 3 → 0xF3

(c) Convert $+47.0_{10}$ to 32-bit IEEE floating Point repres. in binary & hex:

binary: 0100 0010 0011 1100 0000 0000 0000 0000

hex: 0x 4 2 3 C 0 0 0 0

(d)

binary: | 1 | 01111101 | 100... |

Sign | Exp(8) | Fraction (23)

hex: 0xBEC00000

Assignment 1

(Q2) a) a lives on the stack - local var

b) b_ptr lives on the Leap because it is a dynamically ~~stack~~ allocated variable.

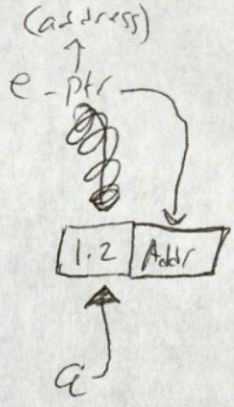
1. c) *b_ptr lives in Heap because b_ptr lives there

d.) e_ptr lives in the Data because it is globally available

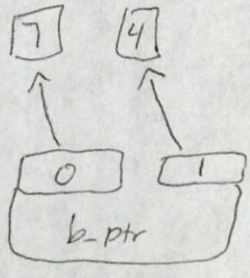
1. e.) *e_ptr lives in Heap, Data because e_ptr lives there.

b) Return value

```
float* e_ptr
float foo(float *x, float *y, float *z){
    if (*x > *y + *z){
        return *x;
    }else{
        return *y + *z;
    }
}
```




```
float a = 1.2
e_ptr = &a;
float *b_ptr = malloc(2 *size of (float));
b_ptr[0] = 7.0;
b_ptr[1] = 4.0;
float c = foo(e_ptr, b_ptr, b_ptr+1);
free(b_ptr);
if(c > 10.5){
    return 0;
}else{
    return 1;
}
}
```



c = foo(e_ptr, b_ptr, b_ptr+1)

c = *y + *z = 4

*e_ptr = *x = 1.2

*b_ptr + *(b_ptr+1) = 0+4

1.2 < 4

or

Function returns 1