Name: _____ Login: _____ IN-CLASS EXERCISE

Struct address syntax

The following questions refer to the code on the right. You may assume that (a) all required #include statements are included, (b) there are no syntax errors, (c) there are no bugs that would prevent the code from running normally. Also, you may assume that sizeof(char) == 1, sizeof(int)==4, sizeof(void*) == 8, and sizeof(double) == 8.

1) Rewrite the following expressions without using the -> operator.

	rewritten without ->
`a_ari -> ale`	
`* a_bev -> bug`	
`& a_bev -> bug`	
`a_bev -> bug + 2`	

```
#pragma pack(1)

typedef struct {
  int ale;
  int amp;
} Ant;

typedef struct {
  double bag;
  char   bib;
  char*  bug;
} Bat;

int main(int argc, char**
argv) {
  Ant ari = {
    .ale = 9,
    .amp = 7
  };
  Ant* a_ari = &ari;

  Bat bev = {
    .bag = 3.7,
    .bib = 'q',
    .bug = "bro"
  };
  Bat* a_bev = &bev;

  int ira[3] = {3, 4};

  return EXIT_SUCCESS;
}
```

Exploring struct objects in memory

2) Underline every local variable. How many did you find? [ ]

3) Double underline every expression that is on the data segment. How many did you find? [ ]

4) Fill in the blanks of the gdb session below.

```
(gdb) b 34
Breakpoint 1 at 0x40055b: file a.c,
line 34.
(gdb) r
Starting program: /…/a

Breakpoint 1, main (argc=1,
argv=0x7fffffffda78) at a.c:34
34      }

(gdb) p sizeof(argc)
$1 = 4
(gdb) [          ] &argc   ⓐ
0x7fffffffd93c: 1
(gdb) x/4bx &argc
0x7fffffffd93c: 0x01    0x00    0x00    0x00
(gdb) x/4bd &argc
0x7fffffffd93c: 1       0       0       0
```

View variables and memory

print[/*format*] *expression*

- *format* ∈ d *(decimal)*, x *(hex)*, s *(string)*, f *(float)*, c *(char.)*, u *(unsigned decimal)*, o *(octal)*, t *(binary)*, z *(zero-padded hex)*
- *expression*: a C expression

x/[*# of units*][*unit*][*format*] *address*

- *# of units*: how many *units*
- *unit* ∈ b *(1 byte)*, h *(2 bytes)*, w *(4 bytes)*, g *(8 bytes)*
- *format*: same as for p/

```
(gdb) p ari
$2 = {ale = 9, amp = 7}
(gdb) p sizeof(ari)
$3 = 8
(gdb) x/4bx &ari
0x7fffffffd970: 0x09    0x00    0x00    0x00
(gdb) x/4bd a_ari
0x7fffffffd970:
```

ⓑ

```
(gdb) x/8bx a_ari
0x7fffffffd970:
```

ⓒ

```
(gdb) x/8bx ira
0x7fffffffd940:
```

ⓓ

```
(gdb) x/2wd ira
0x7fffffffd940:
```

ⓔ

```
(gdb) p bev
$4 = {bag = 3.7000000000000002, bib = 113 'q', bug = 0x400650 "bro"}
(gdb) x/1gx &a_bev -> bug
0x7fffffffd959:
```

ⓕ

```
(gdb) p sizeof(bev)
$5 = 17
(gdb) p bev.bug
$6 = 0x400650 "bro"
(gdb) p &bev.bug[0]
$7 = 0x400650 "bro"
(gdb) p &bev.bug[1]
$8 =
```

ⓖ

```
(gdb) p ira
$9 = {9, 7}
(gdb) x/2wd a_ari
0x7fffffffd970: 9       7
(gdb) p a_ari
$10 = (Ant *) 0x7fffffffd970
```

View variables and memory

print[/*format*] *expression*

- *format* ∈ d *(decimal)*, x *(hex)*, s *(string)*, f *(float)*, c *(char.)*, u *(unsigned decimal)*, o *(octal)*, t *(binary)*, z *(zero-padded hex)*
- *expression*: a C expression

x/[*# of units*][*unit*][*format*] *address*

- *# of units*: how many *units*
- *unit* ∈ b *(1 byte)*, h *(2 bytes)*, w *(4 bytes)*, g *(8 bytes)*
- *format*: same as for p/

```
#pragma pack(1)

typedef struct {
  int ale;
  int amp;
} Ant;

typedef struct {
  double bag;
  char   bib;
  char*  bug;
} Bat;

int main(int argc, char** argv) {
  Ant ari = {
    .ale = 9,
    .amp = 7
  };
  Ant* a_ari = &ari;

  Bat bev = {
    .bag = 3.7,
    .bib = 'q',
    .bug = "bro"
  };
  Bat* a_bev = &bef;

  int ira[3] = {3, 4};

  return EXIT_SUCCESS;
}
```

```
(gdb) x/1gx &a_ari
0x7fffffffd988: [                    ] ⓗ
(gdb) x/8bx &a_ari
0x7fffffffd988: 0x70    0xd9    0xff    0xff    0xff    0x7f    0x00    0x00
(gdb) x/4hx &a_ari
0x7fffffffd988: [                                        ] ⓘ
(gdb) x/2wx &a_ari
0x7fffffffd988: 0xffffd970      0x00007fff
(gdb) x/4bx &a_ari
0x7fffffffd988: 0x70    0xd9    0xff    0xff
(gdb) p &a_ari -> ale
$11 = (int *) [                    ] ⓙ
(gdb) p &a_ari -> amp
$12 = (int *) [                    ] ⓚ
(gdb) [                    ] ①
type = Ant *
(gdb) whatis &a_ari
type = [                    ] ⓜ
(gdb) whatis *a_ari
type = [                    ] ⓝ
(gdb) whatis a_ari[0]
type = [                    ] ⓞ
(gdb) whatis ira
type = int [2]
(gdb) whatis ira[0]
type = [                    ] ⓟ
(gdb)
```

View variables and memory

print[/*format*] *expression*

- *format* ∈ d *(decimal)*, x *(hex)*, s *(string)*, f *(float)*, c *(char.)*, u *(unsigned decimal)*, o *(octal)*, t *(binary)*, z *(zero-padded hex)*
- *expression*: a C expression

x/[*# of units*][*unit*][*format*] *address*

- *# of units*: how many *units*
- *unit* ∈ b *(1 byte)*, h *(2 bytes)*, w *(4 bytes)*, g *(8 bytes)*
- *format*: same as for p/

```
#pragma pack(1)

typedef struct {
  int ale;
  int amp;
} Ant;

typedef struct {
  double bag;
  char   bib;
  char*  bug;
} Bat;

int main(int argc, char** argv) {
  Ant ari = {
    .ale = 9,
    .amp = 7
  };
  Ant* a_ari = &ari;

  Bat bev = {
    .bag = 3.7,
    .bib = 'q',
    .bug = "bro"
  };
  Bat* a_bev = &bef;

  int ira[3] = {3, 4};

  return EXIT_SUCCESS;
}
```