CMSC 216.001 – Spring 2023
Lab 2: Pointers Memory Diagrams
Due: Wednesday, February 22, 11:59PM

1 Introduction

The purpose of this assignment is to increase your familiarity with pointers and memory diagrams. Please submit this on `tpaclinux`, as you did with lab1. The assignment name for the submit program is "lab2". So the resulting submit command line might be: `submit lab2 lab2.pdf`

1. [10 points] Provide declarations for the following variables:
   (a) `p1` a pointer to a generic type or value of unknown type

   void *p1;

   (b) `sp1` a pointer to a string

   char *sp1;

   (c) `dp1` a pointer to a double

   double *dp1;

   (d) `ipp1` a pointer to a pointer to an integer

   int **ipp1;

   (e) `cp1` a character pointer to 80 dynamically allocated characters

   char *cp1 = malloc(80 * sizeof(char));

   (f) `cpp1` a pointer to an array of 80 C-strings

   char (*cpp1)[80];

   (g) `fp1` a pointer to a function that takes two generic pointers and returns an `int`

   int (*fp1)(void *, void *);

   (h) `fp2` a pointer to a function with no parameters or return value

   void (*fp2)(void);

2. [10 points] Given the following code fragment and stack diagram, fill in the name, and contents of each stack location. You may ignore any errors and assume a 64-bit computer, with 32-bit `int`. (You may not need all spaces provided)

```
        int a = 97;
        char *b = a;
        char c = b;
        int data[] = { 95, 96, 97, 98, 99 };
        char *d = &data[2];
        const double E = 2.71828182845904523536;
        int e = (*d)+1;
        int f = *(d+1);
        char g = (*d)+1;
        char h = *(d+1);

```

symbol	value
h	98
g	99
f	99
e	98
E	2.71828182845904523536
d	&data[2]
data[0]	95
data[1]	96
data[2]	97
data[3]	98
data[4]	99
c	'a'
b	&a
a	97

______/10 points

3. [20 points] Consider the following code:

```
void funct(int *a, int *b) {
  int c = *a;
  *a = *b;
  *b = c;

  /**** Point X ****/
}

int main() {
  int x = 2;
  int y = 4;
  int z = 8;

  int *px = &x;
  int *py = &y;
  int *pz = &z;

  /**** Point A ****/
  funct(x, z);

  /**** Point B ****/
  funct(px, py);

  /**** Point C ****/
  funct(&py, &pz);

  /**** Point D ****/
  funct(pz, &px);

  /**** Point E ****/
}
```

Show the stack at:

(a) point A

(b) The first time we reach point X

(c) point B

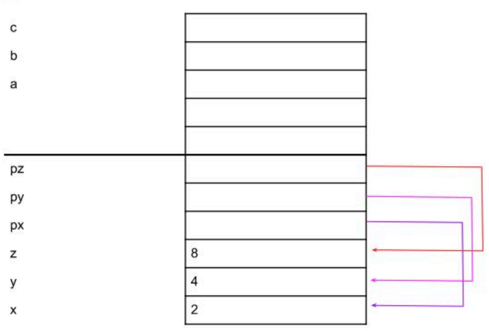
(d) point C

(e) point D

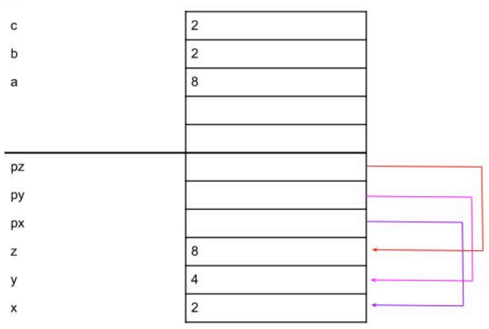
(f) The last time we reach point X

(g) point E

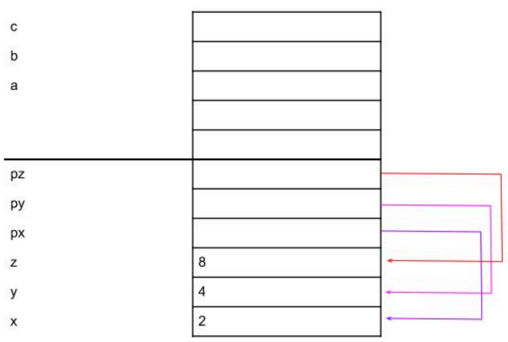
(a)



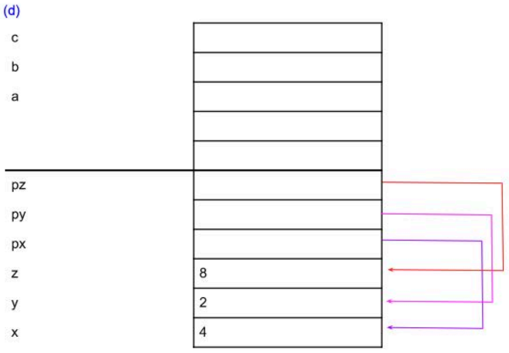
(b)



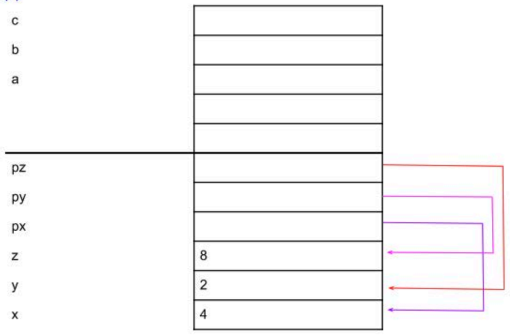
(c)



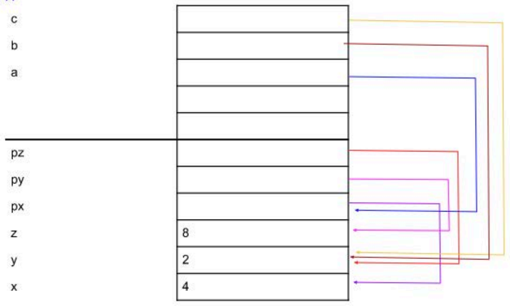
(d)



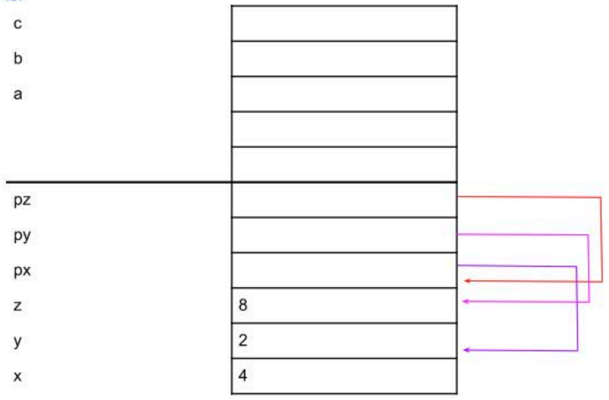
(e)



(f)



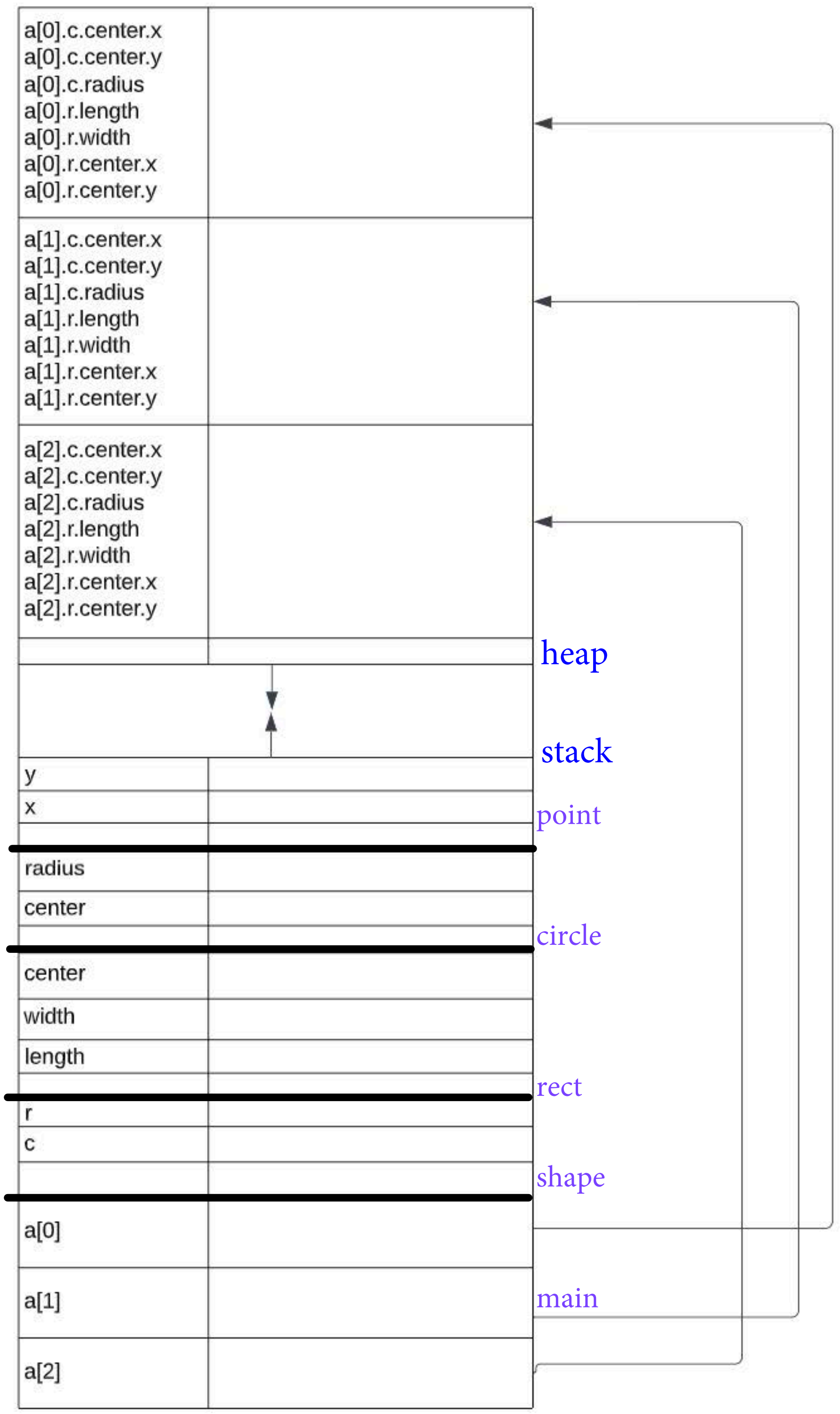
(g)



_______/20 points

4. [10 points] Draw a memory layout & contents diagram for the following code. Please include the contents of the memory as well as the layout! For pointers the contents can just an arrow to the appropriate memory or a virtual NULL memory location.

```
    typedef struct {
      int x;
      int y;
    } point;

    typedef struct {
      point center;
      int radius;
    } circle;

    typedef struct {
      int length;
      int width;
      point center;
    } rect;

    typedef union{
      circle c;
      rect r;
    } shape;

    int main() {
      shape *a = calloc(

      a-> c.center.x = 5
      a-> r.center.x = 6
++;

      a[1].c.center.x =
      a[1].c.center.y =
      a[1].r.center.x =
      a[1].r.center.y=19
    }
```


a[0].c.center.x
a[0].c.center.y
a[0].c.radius
a[0].r.length
a[0].r.width
a[0].r.center.x
a[0].r.center.y
a[1].c.center.x
a[1].c.center.y
a[1].c.radius
a[1].r.length
a[1].r.width
a[1].r.center.x
a[1].r.center.y
a[2].c.center.x
a[2].c.center.y
a[2].c.radius
a[2].r.length
a[2].r.width
a[2].r.center.x
a[2].r.center.y
heap
stack
y
x
point
radius
center
circle
center
width
length
rect
r
c
shape
a[0]
a[1]
main
a[2]