CSC 3320 : System-Level Programming
Fall 2024: HW5 Key

1. (1 point) Ans: Yes, these declarations are legal. We can easily separate them using dot (.) operator. For example, we can use `x.x` or `y.y` to distinguish structure variables and members.
2. (1 point) Ans: 20 bytes (if no padding), 24 bytes (if padding used).
3. (1 point) Ans: 16 bytes without padding, 24 bytes (if padding used: $8 \cup 24 \cup 8 = 24$)
4. (1 point) Ans:
   (a) Ans: Legal and Safe. Valid value assigned to b.
   (b) Ans: Legal, not safe. Integer assigned to b, so it is legal. But i can be any integer, where b is supposed to receive only 0 and 1.
   (c) Ans: legal, not safe. This is for the similar reason as in b).
   (d) Ans: legal and safe. As b is an integer, which can be assigned to any integer variable i.
   (e) Ans: legal and safe. Integer arithmetic will produce integer and can be assigned to an integer variable i.
5. (1 point) Ans:
   (a) `p->b = ' ';` // not legal. Correct `p->d.b`.
   (b) `p->e[3] = 10;` // legal. It uses pointer to access an element of an array member e.
   (c) `(*p).d.a = '*';` // legal. Dereferencing using a pointer (i.e. *p) makes it to behave like an ordinary structure variable. So, we can use dot (.) operator to access individual member.
   (d) `p->d->c = 20;` // not legal, correct `p->d.c`. This is not legal because d is not a pointer. So we cannot use arrow after d.
6. (1 point) Ans: p does not become NULL automatically after `free(p);` But we cannot use it anymore to access data. It becomes a dangling pointer. So, we must preserve the remaining part of the linked list before freeing the current node.

```
struct Node *save;
for (p = first; p != NULL; p = save){
   save=p->next;
   free(p);
}
```

7. (1 point) Ans: It cannot handle insertion at the beginning. Loop test condition is also not properly handled. That is, it must include the terminating condition of the loop, especially when there will be no node left to traverse.

```
struct node *insert_into_ordered_list(struct node *list, struct node *new_node)
{
    if(list == NULL)
        return new_node;
    else if (new_node -> value <= list->value) {
        new_node->next = list;
        return new_node;
    }
    struct node *cur = list, *prev = NULL;
    while (cur != NULL && cur->value <= new_node->value) {
        prev = cur;
        cur = cur->next;
    }
    prev->next = new_node;
```

```
    new_node->next = cur;
return list;
}
```

8. (1 point) Ans: The function can be written as follows:

```
struct node *find_last(struct node *list, int n) {

    struct node *targetNode=NULL;
    struct node *cur = list;
    while (cur != NULL) {
        if(cur->value == n)
            targetNode = cur;
        cur = cur->next;
    }
    return targetNode;
}
```

Question:	1	2	3	4	5	6	7	8	Total
Points:	1	1	1	1	1	1	1	1	8
Bonus Points:	0	0	0	0	0	0	0	0	0
Score:									