

## Unit 2 Sample Problems - C Programming II

In this sample problem set, we will practice concepts of the C programming language.

- Length: 50 minutes with discussion.
- Questions: Q1-Q3, Q5-Q6. (optional: Q4, Q7)

### Memory Allocation

1. [Acuña] Consider the following snippet of code. [4 points total]

```
struct point_2d {
    int x;
    int y;
    Color col;
};

void main() {
    point_2d* data[5];
    for(int i = 0; i < 5; i++) {
        point_2d tmp;
        tmp.x = i;
        tmp.y = i * i;
        data[i] = &tmp;
    }
    for(int i = 0; i < 5; i++)
        printf("Point ##%d: %d, %d", i, data[i]->x, data[i]->y);
}
```

- (a) What will be the output? **Trace this code manually, do not run it.**

- (b) Why does the program give that output? Explain in terms of memory and pointer usage.

2. [Acuña] Consider the following (incorrect) function which removes the head node in a globally defined list called *my\_list*. The struct *grade\_node* is used to represent a node containing grade information in a linked list of grades.

```
struct grade_node {  
    int value;  
    char assignment[255];  
    struct grade_node* next;  
};  
  
struct grade_node* my_list = ...  
  
void remove_node() {  
    free(my_list);  
    if(my_list != null)  
        my_list = my_list->next;  
}
```

(a) What is the syntax error in `remove_node`?

(b) What is the logical error in `remove_node`?

## Defining New Types

3. [Acuña] Consider the problem of padding the following structure, and answer the three questions below. Assume that you are compiling on a system with a 32-bit architecture. [4 points total]

```
struct bmp_header {  
    char creator_name[254];  
    int width;  
    int height;  
    char signature_rgb[2];  
    int offset_pixels;  
};
```

- (a) What is the size of this struct as defined?
- (b) How much space would be wasted with word length padding?
4. [Acuña] One of the issues with using unions is that it can be unclear which element in the union is the one currently being used. Suggest a mechanism to indicate which element is active. [2 points]

## Parameters, Scope, and Pointers

5. [Acuña] Consider two possible functions for adding together two xyz points:

```
struct point add_points(struct point_3d p1, struct point_3d p2)  
struct point* add_points(struct point_3d* p1, struct point_3d* p2)
```

Which of these functions should we expect to operate more efficiently and why?

## Linked Lists

6. [Lisonbee] Using the template provided, **implement** the function 'hide\_nums' that traverses a singly linked list of chars, and for every instance of a number (hint: ASCII values 48-57 correspond to numbers 0-9) it is replaced with a space (ASCII value 32). You can assume that 'sequence' is already initialized with a list of chars, with the address of 'sequence' pointing to the head of the list. Furthermore, the end of the list is represented by a node containing a null terminator.

```
struct char_node {
    char data;
    struct char_node* next;
};

struct char_node* sequence = ...

void hide_nums() {

}
```

7. [Lisonbee] Provide a use case for using an array over a linked list, and one for using a linked list over an array. **Explain** why either data structure is better for each use case.