CSC 3320: System-Level Programming

8 Points

Homework 5 (Extra Credit)

© Instructor: Dr. Md. Mahfuzur Rahman

Fall 2024

Work on the following problems and submit your own answers. You are allowed to discuss with other students. However, do not copy the solutions from peers or other sources. If the assignment has any programming component, your program(s) must compile with **gcc** and execute on **snowball.cs.gsu.edu**! Please see https://cscit.cs.gsu.edu/sp/guide/snowball for more details.

Instructions:

- Upload an electronic copy (MS word or pdf) of your answer sheet to the folder named "HW5" in iCollege.
- Add the course number, homework number, and your name at the top of your answer sheet.
- Write down your answers with the question number only in the answer sheet.
- Use of ChatGPT or other generative models is strictly prohibited. You may receive 0.
- To receive extra credit, submission must be your original work. Your answer/explanation should not match with other students or other sources.
- Name your file in the format of CSC3320_HW5_FisrtnameLastname (.docx/.pdf)
- Deadline: Submit by December 01, 2024, 11:59 pm

1. (1 point) In the following declarations, the x and y structures have members named x and y:

   ```
   struct { int x, y; } x;
   struct { int x, y;} y;
   ```

   Are these declarations legal on an individual basis? Could both declarations appear as shown in a program? Justify your answer.

2. (1 point)
   ```
   struct {
       double a;
       union {
           char b[4];
           double c;
           int d;
       } e;
       char f[4];
   } s;
   ```

   If `char` values occupy one byte, `int` values occupy four bytes, and `double` values occupy eight bytes, how much space will a C compiler allocate for s? Justify. (Assume that the compiler leaves no "holes" between members.) **Hint:** consider the memory alignment requirement for structure/union.

3. (1 point)
   ```
   union {
       double a;
       struct {
           char b[4];
           double c;
   ```

```
        int d;
    } e;
    char f[4];
} u;
```

If `char` values occupy one byte, `int` values occupy four bytes, and `double` values occupy eight bytes, how much space will a C compiler allocate for u? Justify. (Assume that the compiler leaves no "holes" between members.) **Hint:** consider the memory alignment requirement for structure/union.

4. (1 point) Suppose that b and i are declared as follows:

   ```
   enum {FALSE, TRUE} b ;
   int i;
   ```

   Identify which of the following statements are legal and/or "safe' (always yield a meaningful result). Justify.

   (a) `b = FALSE;`

   (b) `b = i;`

   (c) `b++;`

   (d) `i = b;`

   (e) `i = 2 * b + 1;`

5. (1 point) Suppose that f and p are declared as follows:

   ```
   struct {
       union {
           char a b;
           int c;
       } d;
       int e[5];
   } f, *p=&f;
   ```

   Which of the following statements are legal? Why? If not legal, correct them.

   (a) `p->b = ' ';`

   (b) `p->e [3] = 10;`

   (c) `(*p).d.a = '*';`

   (d) `p->d->c = 20;`

6. (1 point) The following loop is supposed to delete all nodes from a linked list and release the memory that they occupy. Unfortunately, the loop is incorrect. Explain what's wrong with it and show how to fix the bug.

   ```
   for (p = first; p != NULL; p = p->next)
      free(p);
   ```

7. (1 point) The following function is supposed lo insert a new node into its proper place in an ordered list, returning a pointer to the first node in the modified list. Unfortunately, the function doesn't work correctly in all cases. Explain what's wrong with it and show how to fix it.

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

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

8. (1 point) Consider the following definition of `node`:

```
    struct node {
         int value;
         struct node *next;
    };
```

Write the following function: `struct node *find_last(struct node *list, int n);` The `list` parameter points to a linked list. The function should return a pointer to the last node that contains n; it should return `NULL` if `n` doesn't appear in the list. Write a complete program to check your function. You **must** submit your complete solution code (.c file).

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:									