Fall 2019 ENEE150 Midterm 2 Practice Solution

Short Answer:

1. What is the advantage of a linked list over an array?

   **Linked List:** prevent memory waste, easier to remove or insert data, dynamic size, don’t have to worry the worst case

   **Array:** Less memory is used if both an array and a linked list try to store the same amount of information, fast to access, random access

2. Write the code that would copy “Lawson is the Best” into array x without wasting any memories.

   ```c
   #include <stdio.h>
   #include <stdlib.h>
   char *x = (int *)malloc(11);
   char y [] = "ENEE205 with Lawson is the BEST";
   
x= (char *)realloc(x,strlen(y+13)+1);
   strcpy(x,y+13);
   ```

   There are other ways to reference those addresses.

3. Assuming you now have “Lawson is the Best” stored in array x, write the code that would print “Best”

   ```c
   printf("%s\n",x+14);
   ```

   Again, there are other ways to reference the address

4. T or F, malloc function is defined in `stdio.h` library and free function is defined in `stdlib.h` library.
   False, both are defined in `stdlib.h`

5. T or F, dynamic allocated data are stored in the heap of the memory
   True

6. What is a possible consequence if the allocated objects are not freed after they are not used?
   Memory Leak

7. Define one struct that can contain information of a course at a university. It should hold the name of the university [100], professor teaching the class [50], course title [50], semester (fall, winter, spring, summer)(seasons are represented by an integer number) , year, number of students, and a data that holds information of all students in the class (assume a struct called student_info is defined). Numbers in square brackets indicate the maximum length of that name.
Solution:

typedef struct class_ {
    char university_name[101];
    char professor[51];
    char title[51];
    int semester;
    int year;
    int num_students;
    struct student_info * student_list;
} class;

8. What are the sizes of follow structures?

typedef struct s1 {
    char* name;
    char class_level[10];
    double gpa;
    int num_credits;
    char major[5];
} info;

typedef struct s2 {
    int number_classes;
    double * class_grades;
    char * class_letter_grades;
    float * final_day;
} academia;

typedef struct s3 {
    struct s1 student_info;
    struct s2 student_academia;
    struct s3 * next;
} student;

48
32
88

https://www.geeksforgeeks.org/is-sizeof-for-a-struct-equal-to-the-sum-of-sizeof-of-each-member/
9. Given:

```c
double a[150];
double *x, *y;
x = a + 50;
y = a + 33;
```

What is \(x - y\)?

10. What is \(*(xyz[1]+5)\), \(*(*(xyz+2)+4)\), \(xyz[1][2]\), \(xyz[1]+5\) ?

6, _ , T , x0083

**Code:**

1. Complete the following function that creates a \(m \times n\) matrix called \(Matrix\);

```c
int **create_matrix(int m, int n){
```
2. Given a sentence (string) that has length N without any punctuations, create a dictionary Dic that stores all words in that sentence, If one word appears more than once, you don’t have to store the repetition. Write a function to create Dic and update size of the dictionary, you may write helper functions. No main is needed. (case doesn’t matter)

```c
char ** make_dic(char *sentence, int N, int *dictionary_size);
```

Solution:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

char ** make_dic(char *sentence, int N, int *dictionary_size);
void tolower_string(char *word);

int main()
{
    char sentence[] = "University of Maryland is a public university in the state of maryland";
    int dictionary_size = 0;
    char ** dictionary = make_dic(sentence,strlen(sentence), &dictionary_size);

    for(i = 0; i<dictionary_size;i++){
        printf("%s\n",dictionary[i]);
    }

    return 0;
}
```

Make_dic:
3. Given a circular linked consist of nodes called Student, write a head insertion function that takes the address of the original head address and the new node to perform this movement. You must update the node that tail node is pointing to, as well as the head.

```c
void head_circular_insertion(pstudent* cicular_list_head, pstudent new_node);
```

Before:

```
  head
     ^
     |     ^
     |     |   |
     |     |   v
     |   v
     v
```

After:

```
  head
     ^
     |     ^
     |     |   |
     |     |   v
     |   v
     v
```
4. You also want to know how to perform **string** manipulation for the exam!!

5. Anything you learned until 10/31 (advanced data structures, recursion, etc.) is a fair game for the exam.