Quiz 1 Review with Sofia F.

Date: Thursday September 12, 2024

Time and location: 4:00 PM at BA1 221

Hello everyone! As some of you may now, I am the SARC SI Leader designated to Dr. Ahmed's section of CS1. In preparation for Quiz 1 I am doing a quiz review. In this document you can find different questions for most of the topics presented in the quiz. You can try these problems on your own, and on the day of the review we will go over the solutions for all of them together.

DISCLAIMER: don't rely solely on this quiz review as preparation for the quiz. Read Dr. Ahmed's announcement thoroughly and make sure you are preparing for everything listed in there.

Dynamic Memory Allocation (DMA)

Dynamic memory functions:

1. Write a single line of code where you use *malloc* to allocate for an array of n integers. Make sure to declare the necessary variables to store the address of the allocated memory.

   _____

2. Write a single line of code where you use *calloc* to allocate for n integers.

   _____

3. Write a single line of code where you use realloc to increase the previously allocated array. Save the reallocated memory appropriately.

   _____

What is the main difference between using malloc and calloc?

How does realloc manage memory?

Fall 2017 **Data Structures Exam, Part A**

1) (5 pts) DSN (Dynamic Memory Management in C)

There is something terribly wrong with the code given below: it has two memory leaks. After carefully inspecting the code, answer the questions below.

```
1:   int main(void)
2:   {
3:     char *str1 = malloc(sizeof(char) * 16);
4:     char *str2 = malloc(sizeof(char) * 16);
5:
6:     str1[0] = 'p';
7:     str1[1] = 'a';
8:     str1[2] = 's';
9:     str1[3] = 's';
10:    str1[4] = ',';
11:    str1[5] = '\0';
12:
13:    printf("%s ", str1);
14:    str2 = str1;
15:    printf("%s ", str2);
16:    str2 = NULL;
17:    strcpy(str1, "pass the exam!");
18:    printf("%s\n", str1);
19:
20:    free(str1);
21:    free(str2);
22:
23:    return 0;
24:  }
```

(a) (3 pts) Draw a picture that indicates the relevant state of memory ***after*** line 14 has completed. (Draw a rectangular box to indicate dynamically allocated memory.)

(b) (1 pt) Explain why line 14 causes a memory leak.

(c) (1 pt) Why is it possible for the code to crash on line 21?

Fall 2016 **Data Structures Exam, Part A**

1) (10 pts) DSN (Dynamic Memory Management in C)

Consider the following struct, which contains a string and its length in one nice, neat package:

```
typedef struct smart_string {
  char *word;
  int length;
} smart_string;
```

Write a function that takes a string as its input, creates a new *smart_string* struct, and stores a **new copy of that string** in the *word* field of the struct and the length of that string in the *length* member of the struct. The function should then return a pointer to that new *smart_string* struct. Use dynamic memory management as necessary. The function signature is:

```
smart_string *create_smart_string(char *str) {
```

Now write a function that takes a *smart_string* pointer (which might be NULL) as its only argument, frees all dynamically allocated memory associated with that struct, and returns NULL when it's finished.

```
smart_string *erase_smart_string(smart_string *s) {
```

```
}
```

Spring 2017 **Data Structures Exam, Part A**

1) (10 pts) DSN (Dynamic Memory Management in C)

A catalogue of ***apps*** and their price is stored in a text file. Each line of the file contains the name of an app (1-19 letters) followed by its price with a space in between. Write a function called ***makeAppArray*** that reads the *app information* from the file and stores it in an array of app pointers. Your function should take 2 parameters: a pointer to the file containing the app information and an integer indicating the number of ***apps*** in the file. It should return a pointer to the array of ***apps***. An ***app*** is stored in a struct as follows:

```
typedef struct{
  char name[20];
  float price;
} app;
```

Make sure to allocate memory dynamically. The function signature is:

```
app** makeAppArray(FILE* fp, int numApps) {
```

Linked List

2) (10 pts) DSN (Linked Lists)

Suppose we have a linked list implemented with the structure below. Write a function that will take in a pointer to the head of a list and inserts a node storing -1 ***after*** each even value in the list. If the list is empty or there are no even values in the list, no modifications should be made to the list. (For example, if the initial list had 2, 6, 7, 1, 3, and 8, the resulting list would have 2, -1, 6, -1, 7, 1, 8, -1.)

```
typedef struct node {
    int data;
    struct node* next;
} node;

void markEven(node *head) {
```

2) (5 pts) DSN (Linked Lists)

Suppose we have a singly linked list implemented with the structure below and a function that takes in the head of the list.

```
typedef struct node {
    int num;
    struct node* next;
} node;

int whatDoesItDo (node * head) {
     struct node * current = head;
     struct node * other, *temp;

     if (current == NULL)
        return head;

     other = current->next;

    if (other == NULL)
        return head;

    other = other->next;
    temp = current->next;
    current->next = other->next;
    current = other->next;

    if (current == NULL) {
        head->next = temp;
        return head;
    }

    other->next = current->next;
    current->next = temp;

    return head;
}
```

If we call whatDoesItDo(head) on the following list, show the list after the function has finished.

head -> 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7

2) (5 pts) ALG (Linked Lists)

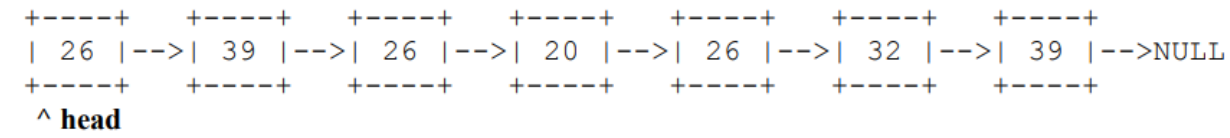
Consider the following function that takes in as a parameter a pointer to the front of a linked list(`list`) and the number of items in the list(`size`). ***node*** is defined as follows:

```
typedef struct node {
    int data;
    struct node* next;
} node;

int mystery(node* list, int size) {
    node* prev = list;
    node* temp = list->next;

    while (temp != NULL) {
        if (list->data == temp->data) {
            prev->next = temp->next;
            free(temp);
            size--;
            temp = prev->next;
        }
        else {
            prev = prev->next;
            temp = temp->next;
        }
    }
    return size;
}
```

If **`mystery(head, 7)`**, is called, where head is shown below, what will the function return and draw a picture of the resulting list, right after the call completes?




```
Adjusted List
-------------
```

Return Value = _____

Base conversion

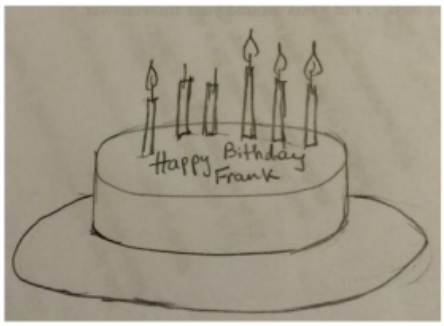
Summer 2015 **Computer Science Exam, Part A**

5) (10 pts) ALG (Base Conversion)

(a) (5 pts) Convert 1654_8 to hexadecimal.

(b) (5 pts) Convert 1925_{10} to octal.

b) (5 pts) Frank is the team-lead for the software testing team at his job. He is celebrating his birthday. Some of his co-workers have baked a cake for the celebration and thought that it would be really cool to put candles on his cake to represent his age in binary. An unlit candle represents the 0 bit. From the pic of the cake below, how old is Max?



_____

Queues

What is the principle for queues?

What are some situations where queues are used in real life?

Write the enqueue and dequeue functions using the following queue struct declaration:

```
typedef struct Queue
{
  int queue_array[MAX];
  int rear;
  int front;
}Queue;
```

```
void enQueue(Queue *q, int add_item)
{









}
```

```
int deQueue(Queue *q)
{









}
```