CPR E 185: Introduction to Computer Engineering and Problem Solving I
Midterm 2: Wednesday (04-12-23)

Last Name: First Name: Lab Section:

1. True/False Questions (10 x 1p each = 10p)

(a) The name of a character array is also a pointer which points to the first element of the array.	TRUE	/	FALSE
(b) The calloc() function reserve memory and initializes to 0	TRUE	/	FALSE
(a) The following statements will have compilation error int x[3] = {2, 3, 9}; int *my_ptr; my_ptr = x[2];	TRUE	/	FALSE

(d) The following printf statement will print 6.
int a[5];
a[-2]=3;
printf("%d", a[-2]);
	TRUE	/	FALSE

(e) char name[15] = "COVID19"; &name[2] is equivalent to name+2	TRUE	/	FALSE
(f) Typedef double usrVariable; usrVariable results; is equivalent to double results;	TRUE	/	FALSE
(g) The following statements will give compilation error: int a[100]; a[100]=6;	TRUE	/	FALSE

(h) The following statements will give compilation error:
int x[3] = {9, 10, 11};
int *my_ptr = x;
printf("%d", *my_ptr);
	TRUE	/	FALSE

(i) The following printf statement prints the address stored inside the pointer variable x. int* x; printf("%p", &x);	TRUE	/	FALSE
(j) The following statement will give a compilation error: char alphabet [4] = {{N},{A},{M},{E}};	TRUE	/	FALSE

2. Pointers (13p)

Given the following block of C code. Write down the contents of **array b** after the following codes are executed. The results are not cumulative (i.e., part (b) is independent of the part (a), etc.). [Show your detail works for the full credits]

(a) (6p)

```
int a[5] = {2, 4, 6, 12, 24};
int b[5] = {16, 10, 9, 12, 7};
int *p, *q, *r;
p = &a[0];
q = &b[0];
while (*q%4!=3){
  *q -= *p++;
  ++q;
}
```

Contents of array b = 11, 10, 9, 12, 7

(b) (7p)

```
int a[5] = {2, 4, 6, 12, 24};
int b[5] = {16, 10, 9, 12, 7};
int *p, *q, *r;
int t;
r = &a[4];
for(p=&a[0], q=&b[0]; p<=r; p++, q++){
   t = *p++; *q = *p++; *p = t+1;
}
for(i=0; i<5; i++){
        printf("%d ",b[i]);
}
```

Contents of array b = 7, 10, 9, 12, 7

3. Struct (20p)

Without altering the program structure, complete the following program that asks the user to enter information for 20 banks, including 100 accounts of each bank. Each bank account info will have the account number, account holder name, and account balance. Finally, your program will print the info regarding every account of 20 banks.

```
#include <stdio.h>
typedef struct{ _____ //(a)fix: add required members to store account info
        char name[50];
        int number[20];
        double balance[50];

}accountInfo;
struct bank {
   char name[50];
   _____//(b):struct variable to store 100 account
};
void main() {
   _____//(c):struct variable to store 20 banks
        for (int i = 0; i < 20; i++) {
                printf("Enter bank%d name: ", i+1);
                scanf("%s", _____); //(d)fix:
                for (int j = 0; j < 100; j++) {
                        printf("\nEnter Bank_%d account_%d information:\n",i+1,j+1);
                        printf("Account number: ");
                        scanf("%d", _____);//(e)fix:
                        printf("Account holder name: ");
                        scanf("%s", _____);//(f)fix:
                        printf("Account balance: ");
                        scanf("%f", _____);//(g)fix:
                }
        }
  for (int i = 0; i < 20; i++) {
          printf("Printing bank%d info: \n", i+1);
          for (int j = 0; j < 100; j++) {
                        printf("\nPrinting Bank_%d account_%d information:\n",i+1,j+1);

                        printf("Account number: %d\n", _____;//(h)fix:

                        printf("Account holder name: %s\n",_____;//(i)fix:

                        printf("Account balance: $%.2f\n", _____;//(j)fix:
                }
  }
}
```

4. Function Output (10p)

Find the output of the following c program. [Note: you must show all the details for full credit]

```
#include <stdio.h>

void fib(int n, int* fib_nums) {
    if (n == 0) {
        fib_nums[0] = 0;
    } else if (n == 1) {
        fib_nums[0] = 0;
        fib_nums[1] = 1;
    } else {
        fib(n-1, fib_nums);
        fib_nums[n] = fib_nums[n-1] + fib_nums[n-2];
    }
}
void main() {
    int fib_nums[10];
    fib(9, fib_nums);
    for (int i = 0; i < 10; i++) {
        printf("%d ", fib_nums[i]);
    }
    printf("\n");
}
```

Answer: 0 1 1 2 3 5 8 13 21 34

5. Array (22p) [Note: You are not allowed to use struct, <u>only a single array</u>]

Let's assume the following table consists of three exam scores of five students. Write a <u>complete c program</u> that will ask the user to enter the following information, and your program will then store the information in **<u>an array</u>** named **exam_scores.** Finally, your program will read from the array to calculate the following and print the results:

(a) Calculate and print the <u>exam average</u> of each student

(b) Calculate the <u>number of students who got B+</u> and print the result. If the exam average is between 82-to-87, consider the grade as B+.

	Student-1	*Student-2*	*Student-3*	*Student-4*	*Student-5*
Midterm 1 ➡	88.50	100	65	99.50	75.50
Midterm 2 ➡	95	99	45	83.25	78.25
Final ➡	97	78	25	100	88.36

```
array [midterm] [students]

# include <stdio.h>
# include <stdlib.h>
int b_students
int main() {
int array[3][5]
int ...
   for (int i=0; i<3; i++){          [i] = exam   [j] = student
      for (int j=0; j<5; j++){
         printf("enter %d student exam %d", j, i)
         scanf("%d", array[i][j]);
      }
   }

   for (int i=0; j<5; i++) {
      total = array[i][j]
      printf("%d" avg/3)
   for (j=0; j<5 ; j++){
    for (int i=0; i<3; i++){
      total += array[i][j]
   }
   printf("%d", total/3)  ->  if (total/3 ≥ 82 && total/3 ≤ 87){
   total=0                        ++ b_students;
   }                            }
   printf("%d", b_students);
}
```

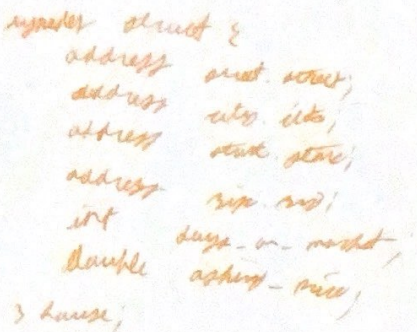
6. Code Snippets (20p)

Define a data structure to store the information for a large real estate agency.

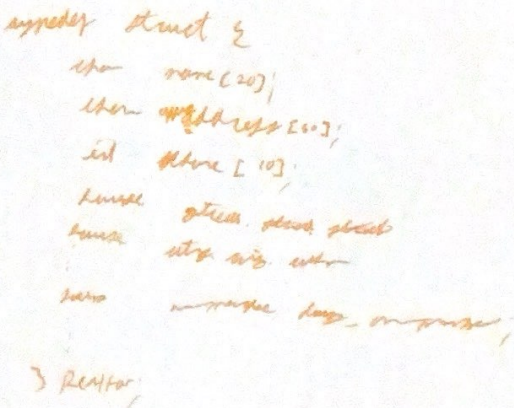
(a) (5p) Write a **struct definition** named **Address** to store the address of any house including street, city, state, and zip code.

```
typedef struct {
    char street[50];
    char city[50];
    char state[2];
    int zip[6];
} address;
```

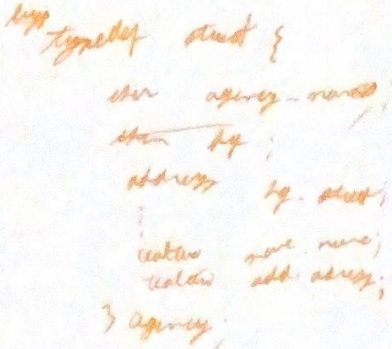
(b) (5p) Write a **struct definition** named **House** to store the information of houses that are listed for sale including **Address**, **number of days** that the house has been on the market, and **asking price**. [note: Address definition is defined in part (a)]



(c) (5p) Write a **struct definition** named **Realtor** to store the information of each Realtor including agent name, office Address, phone number, information of ten houses that are in sale. [note: Address, and House definitions are already defined in part (a) and (b) respectively]



(d) (5p) Write a **struct definition** named **Agency** to store the information of 128 real state agencies including agency name, headquarters address, and the records for up to 100 real estate agents located in different cities. [note: Address, and Realtor definitions are already defined in part (b) and (c) respectively]



7. Struct with function and array (20p)

Without altering the program structure, complete the following program that ask the user to enter 135 student names and their midterm exam score. Then the program prints each student name with their exam score. It will then call a function to calculate the average exam score and print the average score.

```
#include <stdio.h>
#include <stdlib.h>
void sort(int* arr                                                    ) {//(a)fix
   for (int i = 0; i < n-1; i++) {
      for (int j = i+1; j < n; j++) {
         if (arr[i] > arr[j]) {
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
         }
      }
   }
}
void main() {
        int n = 5;
   int *arr = malloc( 5 * sizeof(int));                    ;//(b)fix: allocate dynamic memory
   printf("Enter %d integers: ", n);
   for (int i = 0; i < n; i++) {
      printf("\nEnter element[%d]: ", i);
               scanf("%d", &arr[i]                          );//(c)fix:
   }
   sort(arr, n);
   printf("\nPrinting array after sorting:\n");
   for (int i = 0; i < n; i++                                ) { //(d)fix:print the sorted array


            printf("%d", arr[i]);
   }
   free(arr)                                                  ;//(e)free alocated memory
}
```