ECS 30 **Practice Final Key**

1. (213 points) Write a complete C program that reads information about a house, sorts the information by the area of the rooms, and then displays the information on the screen. Here are the specifications:
   1. On the command line the user will enter the name of the file that contains the information. If there are more command line arguments than required your program must notify the user, and then exit. If the file cannot be found then your program must notify the user, and then exit.
   2. The file has the following format:
      Line 1: Name_of_owner <char [80].>
      Line 2: Price <float> Number_of_rooms <int>
      Lines 3 - 3 + Number_of_rooms: Num_of_windows<int>**:**Area<float>**:**Name of room<char [20]>
   3. The information about each room should be stored in a struct. The information about all of the rooms must be stored in a dynamically allocated array of these structs. You must access the file room lines (lines after the first two) using strtok. Hint: float atof(char* s) will convert a string to a float.
   4. main() must contain only variable declarations and function calls.
   5. The program should have only three functions besides main(): read_file, sort, and show_results. The function named sort will sort the room struct array based on the area of the rooms.
   6. Output of the program should closely match the sample below.

   If the file contains:

```
Bill Mueller
129450.98  4
2:300.5:Living Room
1:107.0:Bedroom 1
2:158.3:Master Bedroom
3:98.3:Kitchen

then the output would be:

Bill Mueller 4 room house  $129450.98
Area   Windows Room
 98.3      3    Kitchen
107.0      1    Bedroom 1
158.3      2    Master Bedroom
300.5      2    Living Roomstrtok(NULL,":")
```

Pts	
2	#include <stdio.h>
2	#include <string.h>
2	#include <stdlib.h>
2	struct room {
2	int windows;
2	float area;
3	char name[20];
1	};
14	void read_file(struct room **rooms, float *price, char owner[], int *num, int argc, char **argv)
	{
1	FILE *fp;
1	int i;
1	char s[256];
4	if(argc != 2) {
3	puts("Error: Wrong number of arguments");
2	exit(1);
	}

7	if((fp = fopen(argv[1],"r")) == NULL) {
5	printf("Error: Unable to open file %s\n", argv[1]);
2	exit(1);
	}
4	fgets(owner, 80, fp);
6	fscanf(fp, "%f%d\n", price, num);
5	*rooms = (struct room*) malloc(sizeof(struct room) * *num);
4	for(i = 0; i < *num; i++) {
4	fgets(s, 256, fp);
7	(*rooms)[i].windows = atoi(strtok(s,":"));
7	(*rooms)[i].area = atof(strtok(NULL,":"));
7	strcpy((*rooms)[i].name, strtok(NULL,"\n"));
	}
1	fclose(fp);
	}
7	void sort(struct room *rooms, int num) {
3	int i, j, MinIndex;
3	struct room temp;
4	for(i = 0; i < num - 1; i++) {
2	MinIndex = i;
6	for(j = i + 1; j < num; j++)
6	if(rooms[MinIndex].area > rooms[j].area)
2	MinIndex = j;
3	temp = rooms[MinIndex];
4	rooms[MinIndex] = rooms[i];
3	rooms[i] = temp;
	} /* for i */
	}
9	void show_results(struct room *rooms, float price, char *owner, int num) {
1	int i;
10	printf("%s %d room house $%.2f\n",owner, num, price);
3	puts("Area Windows Room");
4	for(i = 0; i < num; i++)
12	printf("%5.1f %d %s\n", rooms[i].area, rooms[i].windows, rooms[i].name);
	}
6	int main(int argc, char **argv)
	{
3	struct room *rooms;
1	float price;
2	char owner[80];
1	int num;

7	read_file(&rooms, &price, owner, &num, argc, argv);
3	sort(rooms, num);
5	show_results(rooms, price, owner, num);
2	return 0
	}

2. (10 points) Write the UNIX command that changes the name of a sub-directory from "old_directory_name" to "new_directory_name".
   mv old_directory_name new_directory_name

3. (10 points) In gdb, how would you check the value of a variable var on entry into a function, NewFun( ), and upon exit from it.

There are many ways to do it. One is to setup a breakpoint at the beginning of the function, `b NewFun`, and then run the program using `n` (Next).

4. (50 points) Write a function to delete the last node in a linked list. The header is:

```
NODE * delete_last_node ( NODE * ptr, int * success ) {
NODE * pred;

// SPECIAL CASE 1: nothing to delete
if ( ptr == NULL ) {
    *success = 0;
    return ptr;
}
else {// SPECIAL CASE 2: only one node in list
   if ( ptr->next == NULL ) {
         free(ptr);
         *success = 1;
         return NULL;
   }
else {// there are more than one nodes in the list
  pred = ptr; // save the original pointer
  // find the second last node
  while ( pred->next->next != NULL ) {
         pred = pred->next;
  }
  free ( pred->next );
  pred->next = NULL;
  *success = 1;
  return ptr;
}
```

5. (30 points) How would you copy the values from an array of ints a[] into an array of ints b[] if no loops were allowed?

   You can do it with structs wrapped around the arrays, e.g.:

```
struct as {
   int a [10];
} aa;

struct bs {
   int b[10];
} bb;

…
bb = aa;
```

6. (20 points) Write on the lines provided the output to the screen when following correct program is run. There may be more lines provided than are needed.

```
#include <stdio.h>
#define ADD(x,y)  x + y
#define N 12
main(){
#ifdef N
 puts("Hello");
#else
 puts("Hi");
#endif
  printf("%d\n", ADD(7, 3) * 4);
#undef N
#ifdef N
 puts("Tata");
#else
 puts("Goodbye");
#endif
}
```

__Hello_____

___19_____

___Goodbye_____

_____

_____

_____