ECE 551
Midterm Exam

Name: NetID:

There are 7 questions, with the point values as shown below. You have 75 minutes with a total of 75 points. Pace yourself accordingly.

This exam must be individual work. You may not collaborate with your fellow students. However, this exam is open book and open notes. You may use any printed materials, but no electronic nor interactice resources.

I certify that the work shown on this exam is my own work, and that I have neither given nor received improper assistance of any form in the completion of this work.

Signature:

#	**Question**	**Points Earned**	**Points Possible**
1	Multiple Choice		8
2	Tracing C Code		12
3	Algorithmic Basics		12
4	4: Dynamic Allocation		10
5	Coding 1		11
6	Coding 2		11
7	Recursion		11
	Total		75
	Percent		100

Question 1 Multiple Choice [8 pts]

For each of the following questions, circle the **best** answer:

1. Conceptually, what does the unary `&` operator mean in C?

   (a) Follow the arrow.

   (b) Give an arrow pointing at something.

   (c) Allocate memory for something.

   (d) Concatenate two strings.

   (e) None of these.

2. We compile with `-Wall -Werror` because...

   (a) they make the compiler wall off erroneous code.

   (b) they make the compilation process faster.

   (c) they make the compiler check for errors in all files.

   (d) it is better to have the compiler detect potential errors, and then fix them before running the program.

   (e) None of these.

3. The `fgetc` function...

   (a) ...reads characters from a `FILE *` into an array, up to a specified maximum number of characters.

   (b) ...is inherently unsafe and should not be used.

   (c) ...cannot be used with `stdin` as its stream argument.

   (d) ...allocates memory for one or more characters.

   (e) None of these.

4. How many test cases do you need to be sure a function is correct?

   (a) 42.

   (b) 3 per line of code.

   (c) 3 per decision.

   (d) 3 per path.

   (e) None of these.

Question 2 Tracing C Code [12 pts]

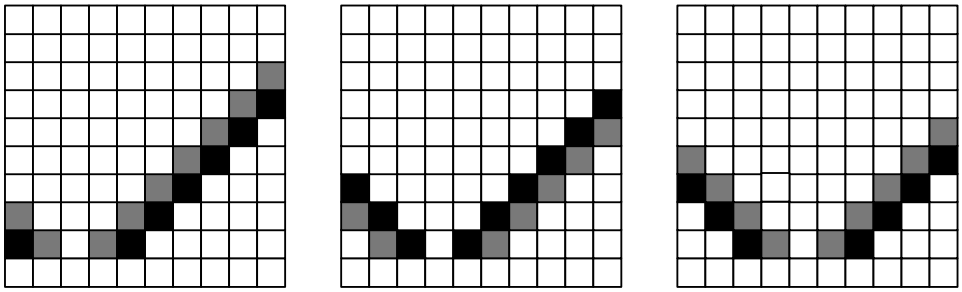
What is the output when the following C code is executed?

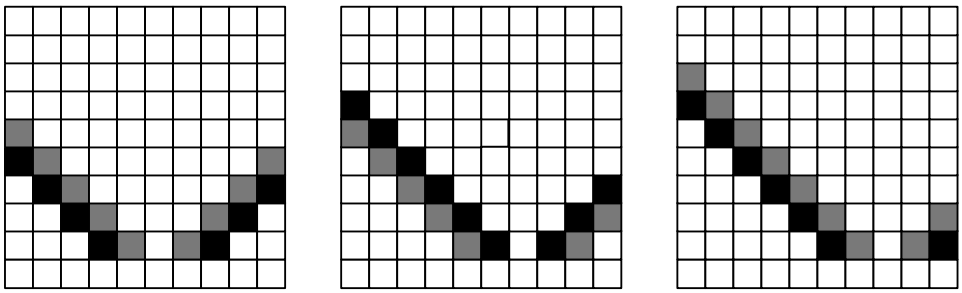
```
#include <stdio.h>
#include <stdlib.h>


int main (void) {
  int a = 3;
  int b = 4;
  int data[3] = {6, 8, 12};
  int * ptrs[3] = {data, &a, &b};
  ptrs[0]++;
  *ptrs[1] += *ptrs[2] ;
  ptrs[0][1]--;
  for (int i = 0; i < 3; i++) {
    printf("*ptrs[%d] = %d\n", i, *ptrs[i]);
  }
  while(data[1] < data[2]) {
    for (int i = 0; i < 3; i++) {
      *ptrs[i] += (i+2);
    }
  }
  printf("data[1] = %d\n", data[1]);
  printf("a = %d\n", a);
  printf("b = %d\n", b);
  return EXIT_SUCCESS;
}
```

Your Output:

Question 3 Algorithmic Basics [12 pts]

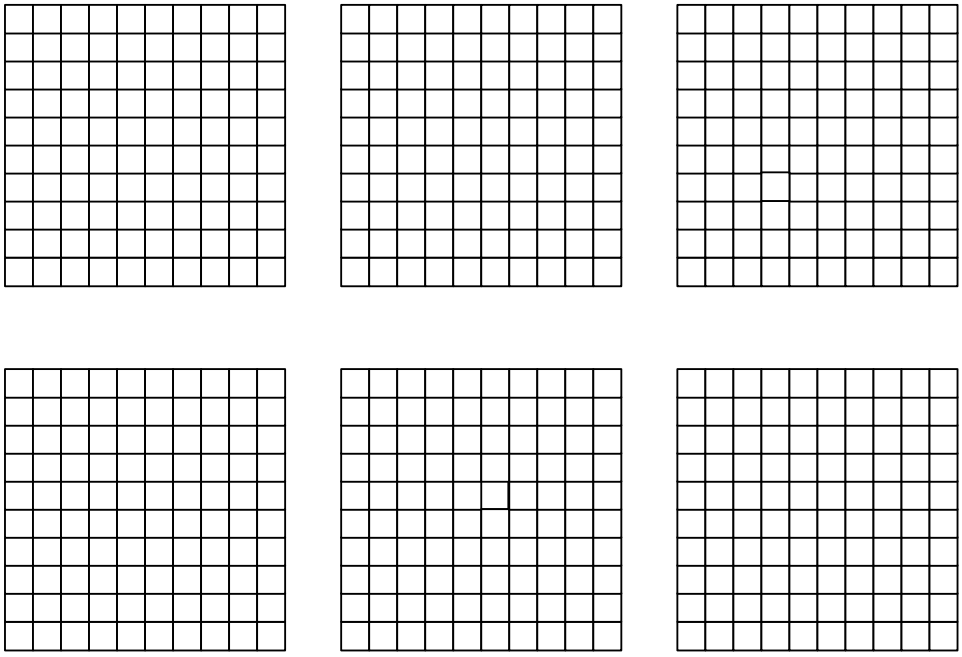
The diagrams shown below are the result of executing an algorithm (parameterized by N) which draws black squares, grey squares, and grey Xes on a 10x10 grid of white boxes.





Determine the algorithm used to create these diagrams, and write the algorithm below in clear-step-by-step English for any N. Note that the next page has blank grids and space for scratch work. Your coordinates should start from (0,0) in the bottom left.

This page is for scratch work. It will not be graded.



This page is for scratch work. It will not be graded.

Question 4 Dynamic Allocation [10 pts]

Consider the following C code (which has errors in it):

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  struct _string_pair_t {
6    char * str1;
7    char * str2;
8  };
9  typedef struct _string_pair_t string_pair_t;
10
11 int main(void) {
12   char * line = NULL;
13   size_t sz;
14   string_pair_t ** array;
15   int count = 0;
16   while (getline(&line, &sz, stdin) >= 0) {
17     array = realloc(array, (count+1) * sizeof(*array));
18     array[count] = malloc(sizeof(array[count]));
19     array[count]->str1 = strdup(line);
20     char * p = strchr(array[count]->str1, '=');
21     if (p != NULL) {
22       *p = '\0';
23       array[count]->str2 = strdup(strchr(line, '=') + 1);
24     }
25     else {
26       array[count]->str2 = NULL;
27     }
28     count++;
29   }
30   for (int i = 0; i < count; i++) {
31     printf("%s = %s\n", array[i]->str1, array[i]->str2);
32     free(array[i]->str1);
33     free(array[i]->str2);
34   }
35   free(array);
36   return EXIT_SUCCESS;
37 }
```

`valgrind` reports the following information about this program (which you may find useful in helping you find the problems):

```
Conditional jump or move depends on uninitialised value(s)
   at 0x4C2CE3B: realloc (in ...)
   by 0x400768: main (q4.c:17)

Invalid write of size 8
   at 0x400847: main (q4.c:26)
 Address 0x51fd158 is 0 bytes after a block of size 8 alloc'd
   at 0x4C2AB80: malloc (in ...)
   by 0x400793: main (q4.c:18)

Invalid write of size 8
   at 0x40082A: main (q4.c:23)
 Address 0x51fd248 is 0 bytes after a block of size 8 alloc'd
   at 0x4C2AB80: malloc (in ...)
   by 0x400793: main (q4.c:18)

Invalid read of size 8
   at 0x400899: main (q4.c:31)
 Address 0x51fd158 is 0 bytes after a block of size 8 alloc'd
   at 0x4C2AB80: malloc (in ...)
   by 0x400793: main (q4.c:18)

Invalid read of size 8
   at 0x400902: main (q4.c:33)
 Address 0x51fd158 is 0 bytes after a block of size 8 alloc'd
   at 0x4C2AB80: malloc (in ...)
   by 0x400793: main (q4.c:18)

HEAP SUMMARY:

XX bytes in YY blocks are definitely lost
   at 0x4C2AB80: malloc (in ...)
   by 0x400793: main (q4.c:18)

XX bytes in YY blocks are definitely lost
   at 0x4C2AB80: malloc (in ...)
   by 0x4EA6AA4: getdelim (iogetdelim.c:66)
   by 0x40086C: main (q4.c:16)
```

Identify and fix the four errors in the code. For each error (1) state the line number that the error is on, (2) state the problem with this line (reason it is an error), and (3) state the correction that fixes this error:

1. Error 1
    (a) Line Number:

    (b) Problem:

    (c) Fix:

2. Error 2
    (a) Line Number:

    (b) Problem:

    (c) Fix:

3. Error 3
    (a) Line Number:

    (b) Problem:

    (c) Fix:

4. Error 4
    (a) Line Number:

    (b) Problem:

    (c) Fix:

Question 5 Coding 1 [11 pts]

Write the function `addEntry` which takes an `info_array_t` (`data`), grows its `data` array so that it can accommodate a new entry, and makes a *deep* copy of the `info_t`, `toAdd`. You may find `strdup` useful for this function.

```
struct _info_t {
   const char * name;
   int num;
};
typedef struct _info_t info_t;

struct _info_array_t {
   info_t * array;
   int numEntries;
};

typedef struct _info_array_t info_array_t;

void addEntry (info_array_t * data, info_t * toAdd) {









}
```

Then write a `freeInfoArray` function which frees all memory associated with an `info_array_t`. Note that if we allocate a `info_array_t` with `malloc`, then perform any number of `addEntry` operations on it, we should be able to free all of that memory with one call to `freeInfoArray`.

```
void freeInfoArray(info_array_t * p) {









}
```

Question 6 Coding 2 [11 pts]

Write a program which takes one command line argument (a file name), reads all of the lines of text in the file named by the command line argument, and prints out each line of input exactly as it was read in, except with all of the lowercase letters changed to capital letters (that is, if your program reads a line that says `I have 2 cats`, then it should print `I HAVE 2 CATS`. If the input file is empty (does not contain any lines) then your program should print no output at all.

You may find `getline` and `toupper` useful on this problem (though you may use any standard library functions you wish).

For this problem, we will relax the "no assumptions" restriction by allowing you to assume that (1) a command line argument is provided (2) the file requested exists and is readable (so fopen succeeds) (3) malloc and realloc always succeed (4) fclose succeeds.

Answer on the next page

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

Question 7 Recursion [11 pts]

Write the recursive function `isOrdered`, which checks if the elements of an array of `int`s are in ascending order—*i.e.*, if element 0 is the smallest, then element 1 is the next smallest, and so on. This function takes two parameters, `array`, which is the array whose ordering should be checked, and `n`, which is the size of that array.

You may not use any library functions (`strlen`, `strdup`, `malloc`, etc). You **MUST** use only recursion for this problem. You **may not** use iteration (no `for`, `while`, or `do-while` loops).

```
int isOrdered ( const int * array, int n ) {




}
```