ECE 551D
Fall 2021
Test 2—Version 1

Name: NetID:

There are 5 questions, with the point values as shown below. You have 75 minutes with a total of 40 points. Pace yourself accordingly.

This exam must be individual work. You may not collaborate with your fellow students. However, this exam is open notes, so you may use your class notes, which must be handwritten by you.

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**
1	Concepts	5
2	Reading Code	9
3	Pictionary	5
4	Coding 1	9
5	Coding 2	12
	Total	40

Question 1 Concepts [5 pts]

For all parts of this question, you *must* blacken the circle of the answer you choose. Remember that we can *only* see the region with the circles when grading.

1. If `p` has type `int **` then what type does `*p` have?

   (a) int

   (b) int *

   (c) int **

   (d) int ***

   (e) None of the above

2. Suppose you want to make `q` be an array with 4 elements, each of which is an integer. What would be the *best* way to fill in the ??? in this statement:
   `int * q = malloc(4 * ???);`

   (a) 4

   (b) 8

   (c) `sizeof(int)`

   (d) `sizeof(*q)`

   (e) None of the above

3. A misguided programmer wants to do something to each character in a file. The programmer opens the file, and puts the resulting `FILE *` in a variable called `f`, abstracts out the code they want to perform on each character into another function (called `do_to_each_char`), and writes this loop to read and process each character:

   ```
   while (fgetc(f) != EOF) {
     do_to_each_char(fgetc(f));
   }
   ```

   What incorrect behavior will this code exhibit?

   (a) The program will process every character in the file twice.

   (b) The program will only process every other character in the file.

   (c) The program will incorrectly handle inputs with character 0xFF in them.

   (d) The program does not account for the `'\0'` character at the end of a string.

   (e) None of the above

4. Suppose you were doing the "kvs" assignment and wanted to copy the key portion from the original string. You abstract out a `get_key` function, which correctly checks that `line` actually has an equals sign, declares a variable `k` for the answer, and allocates the correct amount of space by mallocing and assigning the result to `k`. You then write this loop to copy the values from `line` into `k`.

```
char * get_key(const char * line) {
  /* error checking,
     declaration of k,
     and malloc correctly done here */
  int i = 0;
  while(line[i] != '=') {
    k[i] = line[i];
    i++;
  }
  return k;
}
```

What is incorrect about the above code?

(a) There is an off-by-one error in the loop bounds, which will accidentally include the `'='` in the returned key.

(b) There is an off-by-one error in the loop bounds, which will accidentally omit the last character before the `'='` from the returned key.

(c) The code does not properly handle the case where character 0xFF appears in the key portion of `line`.

(d) The code does not properly null-terminate the resulting key.

(e) None of the above

5. In the recitation video where Drew designed blackjack, he initially hardcoded 4 as the value passed to `init_players`, although the game may have a different number of players. This initial hardcoding is an example of:

(a) Test-driven development

(b) Black-box testing

(c) A test scaffold

(d) White-box testing

(e) None of the above

Question 2 Reading Code [9 pts]

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

int * do_stuff(int ** r, int a, int b) {
  **(r + a) = **(r + a + 1);
  *r[b] = *r[b - 1];
  r[a] = NULL;
  r[b + 1] = NULL;
  return r[a + 1];
}

int main(void) {
  int * x = malloc(2 * sizeof(*x));
  *x = 0;
  *(x + 1) = 1;
  int arr[] = {2, 3};
  int ** q = malloc(4 * sizeof(*q));
  for (int i = 0; i < 4; i++) {
    if (i % 2 == 0) {
      q[i] = &arr[i / 2];
    }
    else {
      q[i] = x + i / 2;
    }
    printf("*q[%d] is %d\n", i, *q[i]);
  }
  int * p = do_stuff(q, 0, 2);
  for (int i = 0; i < 4; i++) {
    if (q[i] == NULL) {
      printf("q[%d] is NULL\n", i);
    }
    else {
      printf("*q[%d] is %d\n", i, *q[i]);
    }
  }
  x = NULL;
  free(p);
  free(q[0]);
  free(q);
  return EXIT_SUCCESS;
}
```

Execute the code on the previous page by hand, write the output, and answer the question below.

Your output should be 8 lines long. Please write each line where indicated below:

Output line 1

Output line 2

Output line 3

Output line 4

Output line 5

Output line 6

Output line 7

Output line 8

Does the code make a memory error? If so, blacken the circle naming the error. If not, choose "None of the above."

(a) Memory leak

(b) Double free

(c) Free an address not returned by malloc

(d) Free an address not on the heap

(e) None of the above

Question 3 Pictionary [5 pts]

Recall the program you wrote to play a game of Minesweeper. The board is stored in a `board_t`, the function `addRandomMine` was provided, and you wrote the function `makeBoard`.

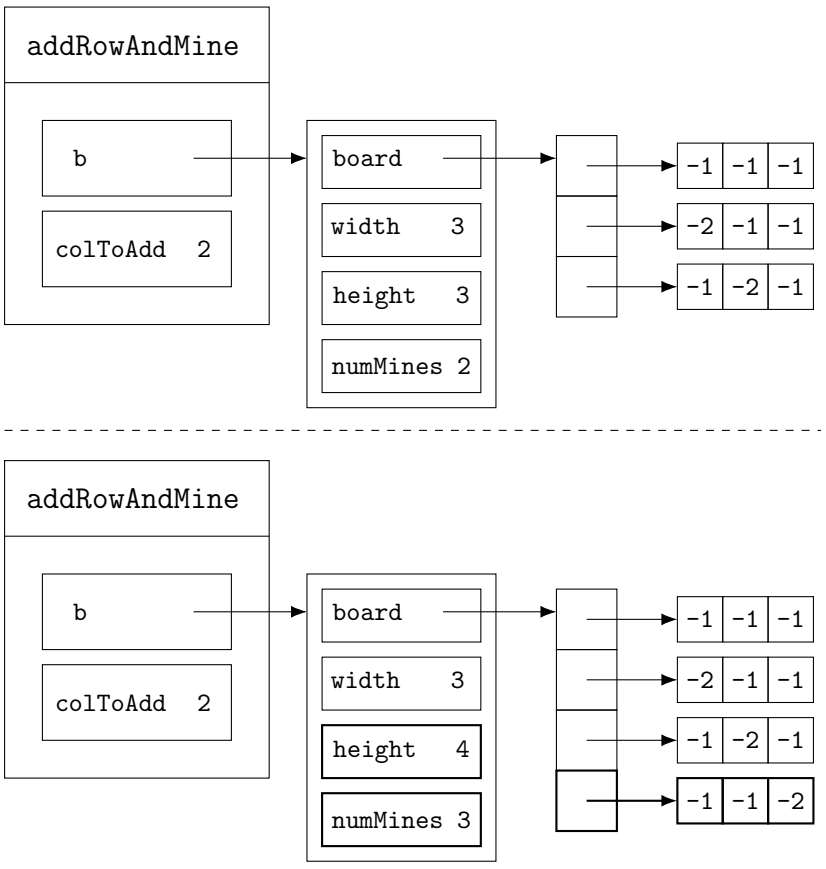
```
#define UNKNOWN -1
#define HAS_MINE -2

struct _board_t {
  int ** board;
  int width;
  int height;
  int numMines;
};
typedef struct _board_t board_t;

void addRandomMine(board_t * b);
board_t * makeBoard(int w, int h, int numMines);
```

Using the picture on the following page, write the function

```
void addRowAndMine(board_t * b, int colToAdd);
```

which should expand the board by one row and add one mine at the column indicated by the second parameter. Your code should change the picture as indicated (changes are outlined with a thick line).

```
void addRowAndMine(board_t * b, int colToAdd) {









}
```


addRowAndMine
b
colToAdd 2
board
width 3
height 3
numMines 2
-1 -1 -1
-2 -1 -1
-1 -2 -1
addRowAndMine
b
colToAdd 2
board
width 3
height 4
numMines 3
-1 -1 -1
-2 -1 -1
-1 -2 -1
-1 -1 -2

Question 4 Coding 1 [9 pts]

Write the function `forwardMaximum`, which takes three parameters: (1) an array of integers `data`, which is the input data, (2) a size_t `n_elements`, which is the number of elements in both `data` and `output`, and (3) an array of integers `output`, which is where you should write the output data.

This function should examine the input data and fill in the output array such that each element of the output array has the largest value of the input array from the corresponding index forwards (meaning to any higher index).

For example, given the following array:
250, 255, 249, 245, 247, 235, 240, 242, 243, 230, 220
This function would fill in the output array with
255, 255, 249, 247, 247, 243, 243, 243, 243, 230, 220.

You may write any helper functions you wish.

```
void forwardMaximum(int * data, size_t n_elements, int * output) {

}
```

Question 5 Coding 2 [12 pts]

For this problem, you are going to use the function you wrote in the previous problem (and assume it works correctly). You are going to write a program that takes one command line argument, the name of a file. The program will read data from the specified file, one integer per line, and then compute the forward maximum of the data that it read. The program will then print out the forward maximum, one number per line. When it is finished, your program should close any files it opened and free any memory it allocated.

For this problem, we will relax the "no assumptions" restriction by allowing you to assume that (1) one command line argument is provided (2) the input file requested exists and is readable (`fopen` succeeds) (3) every line contains a valid integer, which can be correctly converted by `atoi` and fits into an `int` (4) `malloc` and `realloc` always succeed and (5) `fclose` succeeds. You may use any library functions you wish. You may NOT place any restrictions on (nor make assumptions about) the number of lines in the input file.

We will also provide this small function that you may make use of to print your results:

```
void printIntArray(int * data, size_t n) {
  for (size_t i = 0; i < n; i++){
     printf("%d\n", data[i]);
  }
}
```

Please answer on the next page

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

ECE 551D
Fall 2021
Test 2—Version 2

Name: NetID:

There are 5 questions, with the point values as shown below. You have 75 minutes with a total of 40 points. Pace yourself accordingly.

This exam must be individual work. You may not collaborate with your fellow students. However, this exam is open notes, so you may use your class notes, which must be handwritten by you.

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**
1	Concepts	5
2	Reading Code	9
3	Pictionary	5
4	Coding 1	9
5	Coding 2	12
	Total	40

Question 1 Concepts [5 pts]

For all parts of this question, you *must* blacken the circle of the answer you choose. Remember that we can *only* see the region with the circles when grading.

1. If `p` has type `int **` then what type does `&p` have?

   (a) int

   (b) int *

   (c) int **

   (d) int ***

   (e) None of the above

2. Suppose you want to make `q` be an array with 4 elements, each of which is an integer. What would be the *best* way to fill in the ??? in this statement:
   `int * q = malloc(4 * ???);`

   (a) 4

   (b) 8

   (c) `sizeof(int)`

   (d) `sizeof(*q)`

   (e) None of the above

3. A misguided programmer wants to do something to each character in a file. The programmer opens the file, and puts the resulting `FILE *` in a variable called `f`, abstracts out the code they want to perform on each character into another function (called `do_to_each_char`), and writes this loop to read and process each character:

   ```
   while (fgetc(f) != EOF) {
     do_to_each_char(fgetc(f));
   }
   ```

   What incorrect behavior will this code exhibit?

   (a) The program will process every character in the file twice.

   (b) The program will only process every other character in the file.

   (c) The program will incorrectly handle inputs with character 0xFF in them.

   (d) The program does not account for the `'\0'` character at the end of a string.

   (e) None of the above

4. Suppose you were doing the "kvs" assignment and wanted to copy the key portion from the original string. You abstract out a `get_key` function, which correctly checks that `line` actually has an equals sign, declares a variable `k` for the answer, and allocates the correct amount of space by mallocing and assigning the result to `k`. You then write this loop to copy the values from `line` into `k`.

```
char * get_key(const char * line) {
  /* error checking,
     declaration of k,
     and malloc correctly done here */
  int i = 0;
  while(line[i] != '=') {
    k[i] = line[i];
    i++;
  }
  return k;
}
```

What is incorrect about the above code?

(a) There is an off-by-one error in the loop bounds, which will accidentally include the `'='` in the returned key.

(b) There is an off-by-one error in the loop bounds, which will accidentally omit the last character before the `'='` from the returned key.

(c) The code does not properly handle the case where character 0xFF appears in the key portion of `line`.

(d) The code does not properly null-terminate the resulting key.

(e) None of the above

5. In the recitation video where Drew designed blackjack, he initially hardcoded 4 as the value passed to `init_players`, although the game may have a different number of players. This initial hardcoding is an example of:

(a) Test-driven development

(b) Black-box testing

(c) A test scaffold

(d) White-box testing

(e) None of the above

Question 2 Reading Code [9 pts]

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

int * do_stuff(int ** r, int a, int b) {
  **(r + a) = **(r + a - 1);
  *r[b] = *r[b + 1];
  r[a - 1] = NULL;
  r[b] = NULL;
  return r[a + 1];
}

int main(void) {
  int arr[] = {0, 1};
  int * x = malloc(2 * sizeof(*x));
  *x = 2;
  *(x + 1) = 3;
  int ** q = malloc(4 * sizeof(*q));
  for (int i = 0; i < 4; i++) {
    if (i % 2 == 0) {
      q[i] = &arr[i / 2];
    }
    else {
      q[i] = x + i / 2;
    }
    printf("*q[%d] is %d\n", i, *q[i]);
  }
  int * p = do_stuff(q, 1, 2);
  for (int i = 0; i < 4; i++) {
    if (q[i] == NULL) {
      printf("q[%d] is NULL\n", i);
    }
    else {
      printf("*q[%d] is %d\n", i, *q[i]);
    }
  }
  x = NULL;
  free(p);
  free(q[2]);
  free(q);
  return EXIT_SUCCESS;
}
```

Execute the code on the previous page by hand, write the output, and answer the question below.

Your output should be 8 lines long. Please write each line where indicated below:

Output line 1

Output line 2

Output line 3

Output line 4

Output line 5

Output line 6

Output line 7

Output line 8

Does the code make a memory error? If so, blacken the circle naming the error. If not, choose "None of the above."

(a) Memory leak

(b) Double free

(c) Free an address not returned by malloc

(d) Free an address not on the heap

(e) None of the above

Question 3 Pictionary [5 pts]

Recall the program you wrote to play a game of Minesweeper. The board is stored in a `board_t`, the function `addRandomMine` was provided, and you wrote the function `makeBoard`.

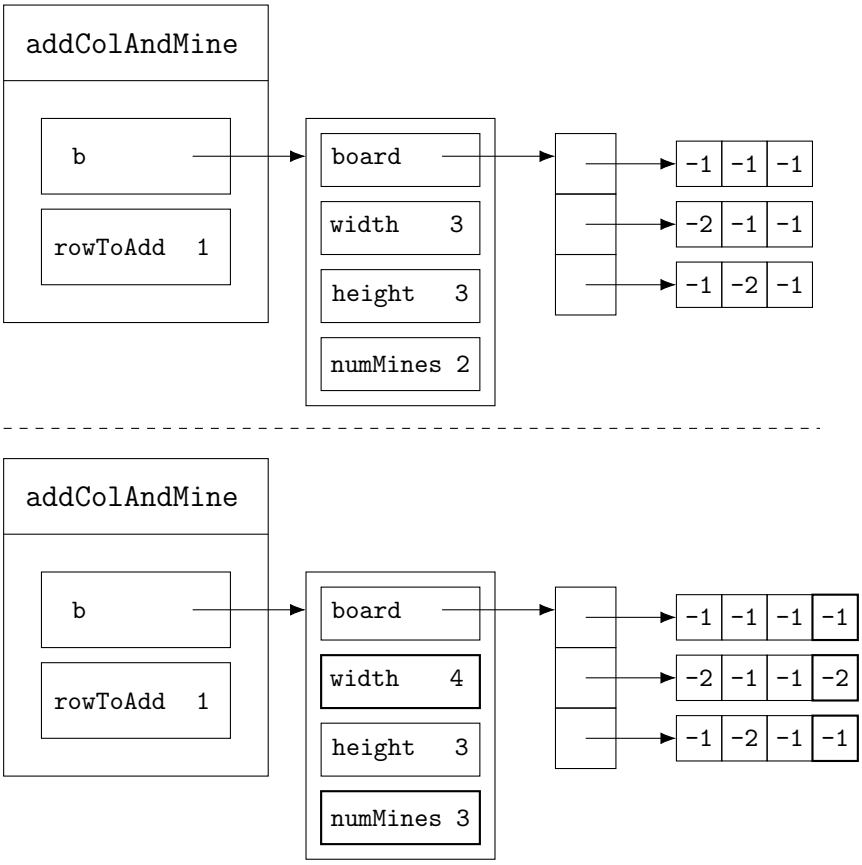
```
#define UNKNOWN -1
#define HAS_MINE -2

struct _board_t {
  int ** board;
  int width;
  int height;
  int numMines;
};
typedef struct _board_t board_t;

void addRandomMine(board_t * b);
board_t * makeBoard(int w, int h, int numMines);
```

Using the picture on the following page, write the function

```
void addColAndMine(board_t * b, int rowToAdd);
```

which should expand the board by one column and add one mine at the row indicated by the second parameter. Your code should change the picture as indicated (changes are outlined with a thick line).

```
void addColAndMine(board_t * b, int rowToAdd) {









}
```


addColAndMine
b
rowToAdd 1
board
width 3
height 3
numMines 2
-1 -1 -1
-2 -1 -1
-1 -2 -1
addColAndMine
b
rowToAdd 1
board
width 4
height 3
numMines 3
-1 -1 -1 -1
-2 -1 -1 -2
-1 -2 -1 -1

Question 4 Coding 1 [9 pts]

Write the function `backwardMinimum`, which takes three parameters: (1) an array of integers `data`, which is the input data, (2) a size_t `n_elements`, which is the number of elements in both `data` and `output`, and (3) an array of integers `output`, which is where you should write the output data.

This function should examine the input data and fill in the output array such that each element of the output array has the smallest value of the input array from the corresponding index backwards (meaning to any lower index).

For example, given the following array:
250, 255, 249, 245, 247, 235, 240, 242, 243, 230, 220
This function would fill in the output array with
250, 250, 249, 245, 245, 235, 235, 235, 235, 230, 220.

You may write any helper functions you wish.

```
void backwardMinimum(int * data, size_t n_elements, int * output) {

}
```

Question 5 Coding 2 [12 pts]

For this problem, you are going to use the function you wrote in the previous problem (and assume it works correctly). You are going to write a program that takes one command line argument, the name of a file. The program will read data from the specified file, one integer per line, and then compute the backward minimum of the data that it read. The program will then print out the backward minimum, one number per line. When it is finished, your program should close any files it opened and free any memory it allocated.

For this problem, we will relax the "no assumptions" restriction by allowing you to assume that (1) one command line argument is provided (2) the input file requested exists and is readable (`fopen` succeeds) (3) every line contains a valid integer, which can be correctly converted by `atoi` and fits into an `int` (4) `malloc` and `realloc` always succeed and (5) `fclose` succeeds. You may use any library functions you wish. You may NOT place any restrictions on (nor make assumptions about) the number of lines in the input file.

We will also provide this small function that you may make use of to print your results:

```
void printIntArray(int * data, size_t n) {
  for (size_t i = 0; i < n; i++){
     printf("%d\n", data[i]);
  }
}
```

Please answer on the next page

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