ECE 551D
Fall 2023
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 45 points. Pace yourself accordingly.

This exam must be individual work. You may not collaborate with your fellow students. However, you are permitted one page of notes.

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	10
3	Debugging	8
4	Coding 1	10
5	Coding 2	12
	Total	45

Question 1 Concepts [5 pts]

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

1. Say you were writing a new string library function `strmult` that copies the string `src` into the buffer `dest` a number `n` times and returns a pointer to `dest`.

   ```
   char * strmult(char * dest, const char * src, size_t n);
   ```

   For example, `strmult(line, "yo", 3);` would copy the string "yoyoyo" into the buffer pointed to by `line`.

   What is the minimum number of bytes `dest` must have?

   (a) n + 1

   (b) strlen(src) * n

   (c) strlen(src) * n + 1

   (d) (strlen(src) + 1) * n

2. Consider an array of strings declared:

   ```
   char * const * strs;
   ```

   Which one of the following is NOT allowed by the const?

   (a) `strs++`

   (b) `(&strs)++`

   (c) `strs[0] = strs[1]`

   (d) `strs[0][0] = 'J'`

3. Recall that one step of your KVs assignment was writing a "counts" data structure and functions to create, add to, and print the counts associated with each name. We provided a main function that hardcoded some data like "apple", "banana", and NULL and called your functions with this data.

   Which one of the following best describes this approach?

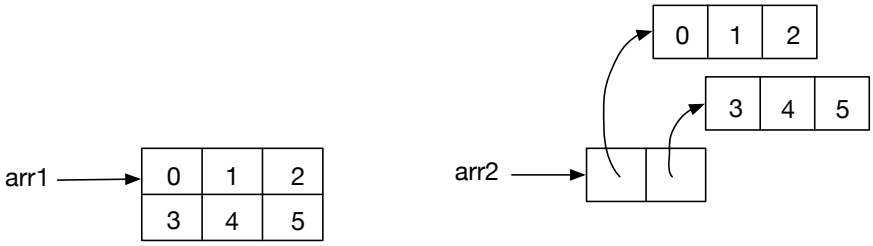
   (a) Test scaffold

   (b) Test-driven development

   (c) Incremental development

   (d) Abstraction

4. Consider the following picture of two different 2D array representations:



Which one of the following is *true* of these pictures?

(a) `arr1[0]` is an lvalue

(b) `arr2[0]` is an lvalue

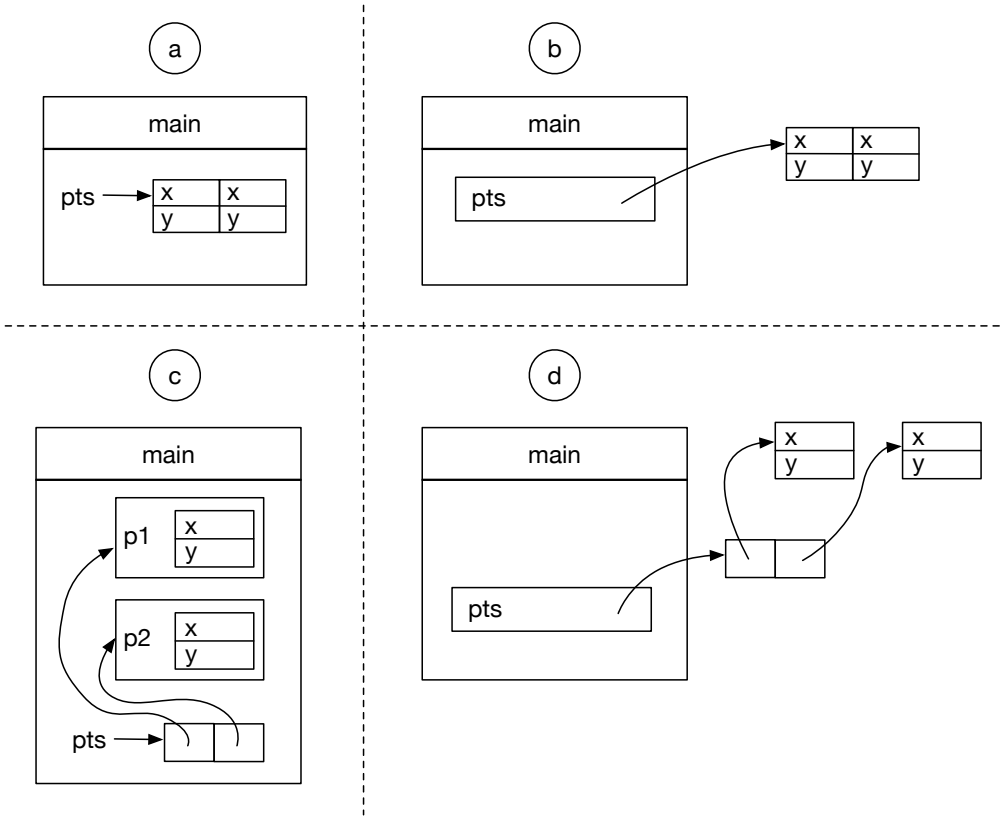
(c) `arr1` represents a 2×3 matrix, while `arr2` represents a 3×2 matrix

(d) `sizeof(arr1)` equals `sizeof(arr2)`

5. Assume you have a struct representing a point with x and y coordinates `point_t`.

Which one of the following pictures correctly shows the variable `pts` when it is declared:

```
point_t * pts[2];
```



Question 2 Reading Code [10 pts]

What is the output when the following C code is executed? Write your answer on the next page. (Assume appropriate header files have been included.)

```
1  void swap(int ** r1, int ** r2) {
2    int * temp = *r1;
3    *r1 = *r2;
4    *r2 = temp;
5  }
6  void printArray(int ** arr_2D, int n, int m) {
7    for (int i = 0; i < n; i++) {
8      printf("%d: ", i);
9      for (int j = 0; j < m; j++) {
10       printf("%d ", arr_2D[i][j]);
11     }
12     printf("\n");
13   }
14 }
15 int ** weirdFunction(int ** q, int n, int m) {
16   swap(q, q + 2);
17   printArray(q, n, m);
18   for (int i = 0; i < n; i++) {
19     for (int j = 0; j < m; j++) {
20       q[i][j] += (3*i + j);
21     }
22   }
23   return q + 1;
24 }
25 int main(void) {
26   int n = 3;
27   int m = 2;
28   int arr0[] = {5, 17};
29   int arr1[] = {12, 8};
30   int arr2[] = {3, 20};
31   int * arr_2D[] = {arr0, arr1, arr2};
32   printArray(arr_2D, n, m);
33   int ** q = weirdFunction(arr_2D, n, m);
34   printf("*q = {%d, %d}\n", q[0][0], q[0][1]);
35   printArray(arr_2D, n, m);
36   return EXIT_SUCCESS;
37 }
```

Your output should be 10 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
Output line 9
Output line 10

Question 3 Debugging [8 pts]

Consider the following code (which has valgrind errors):

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef struct _complex_int_t {
5    int real;
6    int imag;
7  } complex_int_t;
8
9  complex_int_t * init_complex_vector(size_t sz) {
10   complex_int_t * complex_vector = malloc(sz * sizeof(*complex_vector));
11   for (size_t i = 1; i < sz; i++) {
12     complex_vector[i].real = i;
13     complex_vector[i].imag = i + 2;
14   }
15   return complex_vector;
16 }
17
18 complex_int_t complex_add(complex_int_t num1, complex_int_t num2) {
19   complex_int_t sum;
20   sum.real = num1.real + num2.real;
21   sum.imag = num1.imag + num2.imag;
22   return sum;
23 }
24
25 complex_int_t complex_multiply(complex_int_t num1, complex_int_t num2) {
26   complex_int_t product;
27   product.real = num1.real * num2.real - num1.imag * num2.imag;
28   product.imag = num1.real * num2.imag + num1.imag * num2.real;
29   return product;
30 }
```

```
32 complex_int_t * calc_modulus(complex_int_t * complex_vector, size_t sz) {
33   complex_int_t * result = malloc(sizeof(int));
34   result->real = 0;
35   result->imag = 0;
36   for (size_t i = 0; i < sz; i++) {
37     complex_int_t conj;
38     conj.real = complex_vector[i].real;
39     conj.imag = -complex_vector[i].imag;
40     complex_int_t prod = complex_multiply(complex_vector[i], conj);
41     *result = complex_add(*result, prod);
42   }
43   return result;
44 }
45
46 int main(void) {
47   size_t sz = 123;
48   complex_int_t * complex_vector = init_complex_vector(sz);
49   complex_int_t * modulus = calc_modulus(complex_vector, sz);
50   printf("Complex vector:\n");
51   for (size_t i = 0; i < sz; i++) {
52     printf("(%d,%d)\n", complex_vector[i].real, complex_vector[i].imag);
53     free(&complex_vector[i]);
54   }
55   free(complex_vector);
56   printf("The modulus of this complex vector is %d.\n", modulus->real);
57   return EXIT_SUCCESS;
58 }
```

The program is compiled (ignoring warnings—eek!), and run as follows:

```
valgrind ./myProgram
```

For each of the following questions, assume each problem is fixed in order, and the program is recompiled and run again after each question. Even if one of the fixes you select deletes a line, refer to all lines by their original, printed line number.

1. The first error is

```
==13872== Conditional jump or move depends on uninitialised value(s)
==13872==    at 0x48DAAD6: __vfprintf_internal
==13872==    by 0x48C479E: printf
==13872==    by 0x1093F8: main (valgrinderr.c:52)
==13872== Use of uninitialised value of size 8
==13872==    at 0x48BE2EB: _itoa_word
==13872==    by 0x48D9ABD: __vfprintf_internal
==13872==    by 0x48C479E: printf
==13872==    by 0x1093F8: main (valgrinderr.c:52)
```

   Which one of the following describes how to fix this error?

   (a) Change line 10 to be `complex_int_t * complex_vector = malloc(sz);`

   (b) Change line 10 to be
   `complex_int_t complex_vector[sz] = {.real = 0, .imag = 0};`

   (c) Change line 11 to be `for (size_t i = 0; i < sz; i++) {`

   (d) Change line 11 to be `for (size_t i = 1; i <= sz; i++) {`

2. The next error is

```
==14507== Invalid read of size 4
==14507==    at 0x1093CB: main (valgrinderr.c:52)
==14507==  Address 0x4a8f04c is 12 bytes inside a block of size 984 free'd
==14507==    at 0x484B27F: free
==14507==    by 0x109413: main (valgrinderr.c:53)
==14507==  Block was alloc'd at
==14507==    at 0x4848899: malloc
==14507==    by 0x1091C8: init_complex_vector (valgrinderr.c:10)
==14507==    by 0x109383: main (valgrinderr.c:48)
```

   Which one of the following describes how to fix this error?

   (a) Change line 53 to be `free(complex_vector[i]);`

   (b) Delete line 53

   (c) Change line 53 to be `free(*complex_vector[i]);`

   (d) After line 13, add the line `free(&complex_vector[i]);`

3. The next error is

```
==14903== Invalid write of size 4
==14903==    at 0x1092C4: calc_modulus (valgrinderr.c:35)
==14903==    by 0x10939A: main (valgrinderr.c:49)
==14903==  Address 0x4a8f464 is 0 bytes after a block of size 4 alloc'd
==14903==    at 0x4848899: malloc
==14903==    by 0x1092B1: calc_modulus (valgrinderr.c:33)
==14903==    by 0x10939A: main (valgrinderr.c:49)
```

Which one of the following describes how to fix this error?

(a) Delete line 35

(b) Change line 33 to be `int * result = malloc(sizeof(int));`

(c) Change line 33 to be `complex_int_t * result = malloc(sizeof(result));`

(d) Change line 33 to be
`complex_int_t * result = malloc(sizeof(complex_int_t));`

4. There are no more errors, but the program leaks memory. When run

```
valgrind --leak-check=full ./myProgram
```

valgrind reports:

```
==15229== HEAP SUMMARY:
==15229==     in use at exit: 8 bytes in 1 blocks
==15229==   total heap usage: 3 allocs, 2 frees, 2,016 bytes allocated
==15229==
==15229== 8 bytes in 1 blocks are definitely lost in loss record 1 of 1
==15229==    at 0x4848899: malloc
==15229==    by 0x1092B1: calc_modulus (valgrinderr.c:33)
==15229==    by 0x10939A: main (valgrinderr.c:49)
```

Which one of the following describes how to fix this leak?

(a) After line 42, add the line `free(result);`

(b) After line 49, add the line `free(modulus);`

(c) After line 55, add the line `free(modulus);`

(d) After line 56, add the line `free(modulus);`

Question 4 Coding 1 [10 pts]

Write the function `getEvenMultiples` which takes an `array_t data` and factor `int n`, and returns an `array_t` of numbers in `data` that are evenly divisible by `n`. An `array_t` is defined as

```
struct array_tag {
    int * arr;
    size_t sz;
};
typedef struct array_tag array_t;
```

For example, if your function were given an input of: { `14, 24, 21` } and `n = 7`, it would return the an `array_t` whose `arr` field is a dynamically allocated array { `14, 21` } and whose `sz` field is 2. These values are the answer as 14 and 21 are even multiples of 7, but 24 is not.

For this problem you may assume that `malloc` and `realloc` never fail.

Please answer on the next page

```
array_t getEvenMultiples(array_t data, int n) {









}
```

Question 5 Coding 2 [12 pts]

For this problem, you are going to write a program using the function you wrote in the previous problem (and assume it works correctly). The program should take zero or one command line argument, the name of a file. If there are no command line arguments, the program should read from stdin; otherwise, the program will read data from the specified file.

The format of the file is:

- First line: number of elements in data (call it sz)
- Next sz lines: one number per line
- Last line: factor n

Example format from Question 4:

```
3
14
24
21
7
```

The program should

1. Read the data from stdin or the specified file into an `array_t` and `int` to be used with your function `getEvenMultiples`;
2. Call `getEvenMultiples` from the previous question; and
3. Use our `printData` function to print the values in the `array_t` returned by `getEvenMultiples`.

You must also be sure to free any memory and clean up any other resources you need to in the appropriate places.

Note you should NOT write `printData`. You should assume we have written it already and that it has the following declaration:

```
void printData(array_t data);
```

You may assume that `fopen`, `fclose`, `malloc`, and `realloc` all succeed. You may also assume that exactly zero or one command line arguments are passedand that the data in the file is the correct format. You may NOT assume the length of each line.

For full credit, your answer should use abstraction, such that it does not duplicate code to read from a file or stdin.

Please answer on the next page

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

ECE 551D
Fall 2023
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 45 points. Pace yourself accordingly.

This exam must be individual work. You may not collaborate with your fellow students. However, you are permitted one page of notes.

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	10
3	Debugging	8
4	Coding 1	10
5	Coding 2	12
	Total	45

Question 1 Concepts [5 pts]

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

1. Say you were writing a new string library function `strmult` that copies the string `src` into the buffer `dest` a number `n` times and returns a pointer to `dest`.

   ```
   char * strmult(char * dest, const char * src, size_t n);
   ```

   For example, `strmult(line, "yo", 3);` would copy the string "yoyoyo" into the buffer pointed to by `line`.

   What is the minimum number of bytes `dest` must have?

   (a) n + 1

   (b) strlen(src) * n

   (c) strlen(src) * n + 1

   (d) (strlen(src) + 1) * n

2. Consider an array of strings declared:

   ```
   char * * const strs;
   ```

   Which one of the following is NOT allowed by the const?

   (a) `strs++`

   (b) `(&strs)++`

   (c) `strs[0] = strs[1]`

   (d) `strs[0][0] = 'J'`

3. Recall that one step of your KVs assignment was writing a "counts" data structure and functions to create, add to, and print the counts associated with each name. We provided a main function that hardcoded some data like "apple", "banana", and NULL and called your functions with this data.

   Which one of the following best describes this approach?

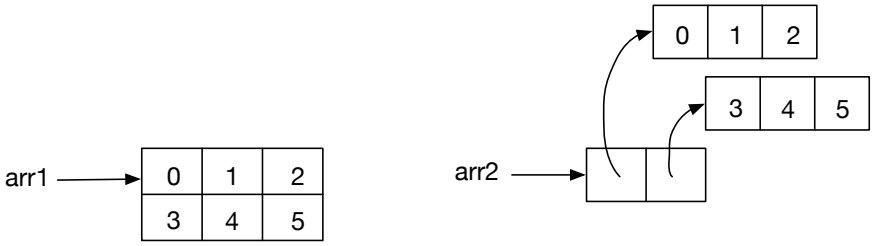
   (a) Test scaffold

   (b) Test-driven development

   (c) Incremental development

   (d) Abstraction

4. Consider the following picture of two different 2D array representations:



Which one of the following is *true* of these pictures?

(a) `arr1[0]` is an lvalue

(b) `arr1` is an lvalue

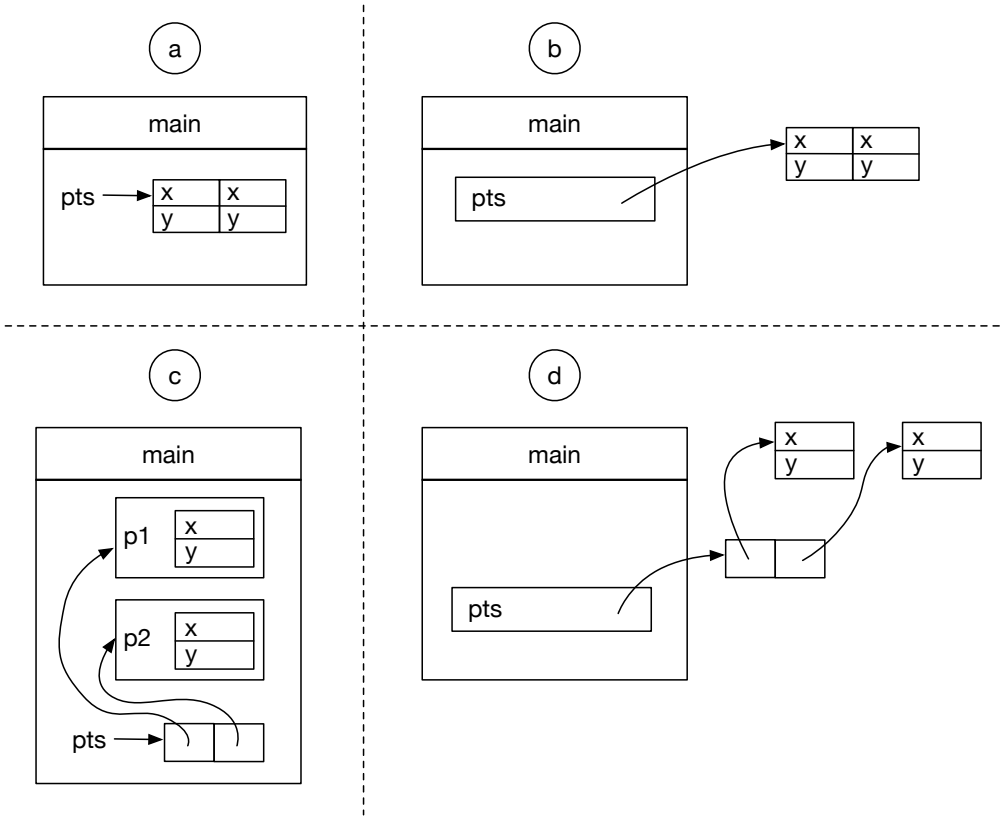
(c) `arr1` represents a 2×3 matrix, while `arr2` represents a 3×2 matrix

(d) `sizeof(arr1)` is different than `sizeof(arr2)`

5. Assume you have a struct representing a point with x and y coordinates `point_t`.

Which one of the following pictures correctly shows the variable `pts` when it is declared:

```
point_t * pts;
```



Question 2 Reading Code [10 pts]

What is the output when the following C code is executed? Write your answer on the next page. (Assume appropriate header files have been included.)

```
1  void swap(int ** r1, int ** r2) {
2    int * temp = *r1;
3    *r1 = *r2;
4    *r2 = temp;
5  }
6  void printArray(int ** arr_2D, int n, int m) {
7    for (int i = 0; i < n; i++) {
8      printf("%d: ", i);
9      for (int j = 0; j < m; j++) {
10       printf("%d ", arr_2D[i][j]);
11     }
12     printf("\n");
13   }
14 }
15 int ** weirdFunction(int ** q, int n, int m) {
16   swap(q, q + 1);
17   printArray(q, n, m);
18   for (int i = 0; i < n; i++) {
19     for (int j = 0; j < m; j++) {
20       q[i][j] += (i + 3*j);
21     }
22   }
23   return q + 2;
24 }
25 int main(void) {
26   int n = 3;
27   int m = 2;
28   int arr0[] = {3, 20};
29   int arr1[] = {11, 5};
30   int arr2[] = {8, 14};
31   int * arr_2D[] = {arr0, arr1, arr2};
32   printArray(arr_2D, n, m);
33   int ** q = weirdFunction(arr_2D, n, m);
34   printf("*q = {%d, %d}\n", q[0][0], q[0][1]);
35   printArray(arr_2D, n, m);
36   return EXIT_SUCCESS;
37 }
```

Your output should be 10 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
Output line 9
Output line 10

Question 3 Debugging [8 pts]

Consider the following code (which has valgrind errors):

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef struct _complex_int_t {
5    int real;
6    int imag;
7  } complex_int_t;
8
9  complex_int_t * init_complex_vector(size_t sz) {
10   complex_int_t * complex_vector = malloc(sz * sizeof(*complex_vector));
11   for (size_t i = 1; i < sz; i++) {
12     complex_vector[i].real = i;
13     complex_vector[i].imag = i + 2;
14   }
15   return complex_vector;
16 }
17
18 complex_int_t complex_add(complex_int_t num1, complex_int_t num2) {
19   complex_int_t sum;
20   sum.real = num1.real + num2.real;
21   sum.imag = num1.imag + num2.imag;
22   return sum;
23 }
24
25 complex_int_t complex_multiply(complex_int_t num1, complex_int_t num2) {
26   complex_int_t product;
27   product.real = num1.real * num2.real - num1.imag * num2.imag;
28   product.imag = num1.real * num2.imag + num1.imag * num2.real;
29   return product;
30 }
```

```
32 complex_int_t * calc_modulus(complex_int_t * complex_vector, size_t sz) {
33   complex_int_t * result = malloc(sizeof(int));
34   result->real = 0;
35   result->imag = 0;
36   for (size_t i = 0; i < sz; i++) {
37     complex_int_t conj;
38     conj.real = complex_vector[i].real;
39     conj.imag = -complex_vector[i].imag;
40     complex_int_t prod = complex_multiply(complex_vector[i], conj);
41     *result = complex_add(*result, prod);
42   }
43   return result;
44 }
45
46 int main(void) {
47   size_t sz = 123;
48   complex_int_t * complex_vector = init_complex_vector(sz);
49   complex_int_t * modulus = calc_modulus(complex_vector, sz);
50   printf("Complex vector:\n");
51   for (size_t i = 0; i < sz; i++) {
52     printf("(%d,%d)\n", complex_vector[i].real, complex_vector[i].imag);
53     free(&complex_vector[i]);
54   }
55   free(complex_vector);
56   printf("The modulus of this complex vector is %d.\n", modulus->real);
57   return EXIT_SUCCESS;
58 }
```

The program is compiled (ignoring warnings—eek!), and run as follows:

```
valgrind ./myProgram
```

For each of the following questions, assume each problem is fixed in order, and the program is recompiled and run again after each question. Even if one of the fixes you select deletes a line, refer to all lines by their original, printed line number.

1. The first error is

```
==13872== Conditional jump or move depends on uninitialised value(s)
==13872==    at 0x48DAAD6: __vfprintf_internal
==13872==    by 0x48C479E: printf
==13872==    by 0x1093F8: main (valgrinderr.c:52)
==13872== Use of uninitialised value of size 8
==13872==    at 0x48BE2EB: _itoa_word
==13872==    by 0x48D9ABD: __vfprintf_internal
==13872==    by 0x48C479E: printf
==13872==    by 0x1093F8: main (valgrinderr.c:52)
```

   Which one of the following describes how to fix this error?

   (a) Change line 10 to be `complex_int_t * complex_vector = malloc(sz);`

   (b) Change line 10 to be
   `complex_int_t complex_vector[sz] = {.real = 0, .imag = 0};`

   (c) Change line 11 to be `for (size_t i = 0; i < sz; i++) {`

   (d) Change line 11 to be `for (size_t i = 1; i <= sz; i++) {`

2. The next error is

```
==14507== Invalid read of size 4
==14507==    at 0x1093CB: main (valgrinderr.c:52)
==14507==  Address 0x4a8f04c is 12 bytes inside a block of size 984 free'd
==14507==    at 0x484B27F: free
==14507==    by 0x109413: main (valgrinderr.c:53)
==14507==  Block was alloc'd at
==14507==    at 0x4848899: malloc
==14507==    by 0x1091C8: init_complex_vector (valgrinderr.c:10)
==14507==    by 0x109383: main (valgrinderr.c:48)
```

   Which one of the following describes how to fix this error?

   (a) Change line 53 to be `free(complex_vector[i]);`

   (b) Delete line 53

   (c) Change line 53 to be `free(*complex_vector[i]);`

   (d) After line 13, add the line `free(&complex_vector[i]);`

3. The next error is

```
==14903== Invalid write of size 4
==14903==    at 0x1092C4: calc_modulus (valgrinderr.c:35)
==14903==    by 0x10939A: main (valgrinderr.c:49)
==14903==  Address 0x4a8f464 is 0 bytes after a block of size 4 alloc'd
==14903==    at 0x4848899: malloc
==14903==    by 0x1092B1: calc_modulus (valgrinderr.c:33)
==14903==    by 0x10939A: main (valgrinderr.c:49)
```

Which one of the following describes how to fix this error?

(a) Delete line 35

(b) Change line 33 to be `int * result = malloc(sizeof(int));`

(c) Change line 33 to be `complex_int_t * result = malloc(sizeof(result));`

(d) Change line 33 to be
`complex_int_t * result = malloc(sizeof(complex_int_t));`

4. There are no more errors, but the program leaks memory. When run

```
valgrind --leak-check=full ./myProgram
```

valgrind reports:

```
==15229== HEAP SUMMARY:
==15229==     in use at exit: 8 bytes in 1 blocks
==15229==   total heap usage: 3 allocs, 2 frees, 2,016 bytes allocated
==15229==
==15229== 8 bytes in 1 blocks are definitely lost in loss record 1 of 1
==15229==    at 0x4848899: malloc
==15229==    by 0x1092B1: calc_modulus (valgrinderr.c:33)
==15229==    by 0x10939A: main (valgrinderr.c:49)
```

Which one of the following describes how to fix this leak?

(a) After line 42, add the line `free(result);`

(b) After line 49, add the line `free(modulus);`

(c) After line 55, add the line `free(modulus);`

(d) After line 56, add the line `free(modulus);`

Question 4 Coding 1 [10 pts]

Write the function `getEvenMultiples` which takes an `array_t data` and factor `int n`, and returns an `array_t` of numbers in `data` that are evenly divisible by `n`. An `array_t` is defined as

```
struct array_tag {
    int * arr;
    size_t sz;
};
typedef struct array_tag array_t;
```

For example, if your function were given an input of: { `14, 24, 21` } and `n = 7`, it would return the an `array_t` whose `arr` field is a dynamically allocated array { `14, 21` } and whose `sz` field is 2. These values are the answer as 14 and 21 are even multiples of 7, but 24 is not.

For this problem you may assume that `malloc` and `realloc` never fail.

Please answer on the next page

```
array_t getEvenMultiples(array_t data, int n) {









}
```

Question 5 Coding 2 [12 pts]

For this problem, you are going to write a program using the function you wrote in the previous problem (and assume it works correctly). The program should take zero or one command line argument, the name of a file. If there are no command line arguments, the program should read from stdin; otherwise, the program will read data from the specified file.

The format of the file is:

- First line: number of elements in data (call it sz)
- Next sz lines: one number per line
- Last line: factor n

Example format from Question 4:

```
3
14
24
21
7
```

The program should

1. Read the data from stdin or the specified file into an `array_t` and `int` to be used with your function `getEvenMultiples`;
2. Call `getEvenMultiples` from the previous question; and
3. Use our `printData` function to print the values in the `array_t` returned by `getEvenMultiples`.

You must also be sure to free any memory and clean up any other resources you need to in the appropriate places.

Note you should NOT write `printData`. You should assume we have written it already and that it has the following declaration:

```
void printData(array_t data);
```

You may assume that `fopen`, `fclose`, `malloc`, and `realloc` all succeed. You may also assume that exactly zero or one command line arguments are passedand that the data in the file is the correct format. You may NOT assume the length of each line.

For full credit, your answer should use abstraction, such that it does not duplicate code to read from a file or stdin.

Please answer on the next page

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