

# CMPT 125 D200, Spring 2023

## Midterm Exam March 3, 2023

Name \_\_\_\_\_

SFU ID: |\_|\_|\_|\_|\_|\_|\_|\_|\_|\_|

Problem 1	
Problem 2	
Problem 3	
TOTAL	

### Instructions:

1. Duration of the exam is 90 minutes.
2. Write your full name and SFU ID **\*\*clearly\*\***.
3. This is a closed book exam, no calculators, cell phones, or any other material.
4. The exam consists of three (3) problems.
5. Write your answers in the provided space.
6. There is an extra page at the end of the exam. You may use it if needed.
7. Explain all your answers.
8. Really, explain all your answers.

Good luck!

### Problem 1 [40 points]

a) [10 points] What will be the output of the following program? Explain your answer.

```
#include <stdio.h>
const int RED = 0;
const int GREEN = 1;
const int BLUE = 2;
const int YELLOW = 3;

int fun(int* x, int* y) {
    long z = 4;
    *y = z;
    z = 65;
    x = y;
    *x = YELLOW;
    return *y;
}

int main() {
    int a = RED, b = GREEN;
    int c = fun(&a, &b);
    printf("a = %d, b = %d, c = %d\n", a, b, c);
    return 0;
}
```

b) [10 points] Will the code below compile?

If yes, what will be the result of the execution? If not, explain errors/warnings/potential issues.

```
#include <stdio.h>
int main() {
    char str[20] = {0, 'A', 'B', 0, '4', '5', '6', '7', 0};
    int ind=0;
    while (str[ind])
        ind++;

    printf("%s\n", str+5);
    return 0;
}
```

c) [10 points] Will the code below compile?

If yes, what will be the result of the execution? If not, explain errors/warnings/potential issues.

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

int* what(int start) {
    int* arr = malloc(4*sizeof(int));
    for(int i=0;i<4;i++)
        arr[i] = start+i*i;
    return arr;
}

int main() {
    int* a1 = what(0);
    a1[0] = 200;
    int* a2 = what(1);
    a2 = a1;
    printf("a1 = [%d, %d, %d, %d]\n", a1[0], a1[1], a1[2], a1[3]);
    printf("a2 = [%d, %d, %d, %d]\n", a2[0], a2[1], a2[2], a2[3]);
    free(a2);
    free(a1);
    printf("return\n");
    return 0;
}
```

d) [10 points] Will the code below compile?

If yes, what will be the result of the execution? If not, explain errors/warnings/potential issues.

```
#include <stdio.h>

int array_sum(int arr[], int length) {
    arr[0]=0; // do something unexpected
    return 0;
}

int main() {
    const int arr[] = {0,0,0,0};
    array_sum(arr, 3);
    return 0;
}
```

## Problem 2 [30 points]

a) [15 points] Write a function that gets an int  $n \geq 0$  and returns a string containing this int. Make sure the returned string is allocated on the heap.

For example,

- `int2str(10)` returns `"10"`.
- `int2str(4)` returns `"4"`.
- `int2str(0)` returns `"0"`.
- `int2str(625)` returns `"625"`.
- `int2str(88088)` returns `"88088"`.

```
char* int2str(unsigned int n) {
```

```
}
```

b) [15 points] Write a function that gets an array of non-negative ints of length  $n > 0$ , and returns a string representing this array in the following format:

- `array_to_string([2,123,103], n=3)` returns "`[2,123,103]`".
- `array_to_string([22,3,40,100], n=4)` returns "`[22,3,40,100]`".
- `array_to_string([41], n=1)` returns "`[41]`".

No spaces between numbers and commas

Make sure the returned string is allocated on the heap.

Remember to free all unnecessary intermediate data allocated on the heap.

You may use the standard functions if necessary (e.g., use the library `string.h`).

```
char* array_to_string(const int* arr, int n) {
```

```
}
```

### Problem 3 [30 points]

a) [15 points] Write a function that gets an int  $n$  and returns a string of length  $n$ , of the form "01234567890123...".

That is, the  $i$ 'th character in the string (counting from zero) is the unit digit of  $i$ .

For example,

- `get_string(4)` returns "0123".
- `get_string(0)` returns "".
- `get_string(10)` returns "0123456789".
- `get_string(12)` returns "012345678901".
- `get_string(25)` returns "0123456789012345678901234".

Make sure the returned string is allocated on the heap.

You may use the standard functions if necessary (e.g., use the library `string.h`).

```
char* get_string(int n) {
```

b) [15 points] Write a function that gets an int  $n > 0$  and returns an array of strings of length  $n$ , where the  $i$ 'th string is `get_string(i)` from the previous item. You may assume  $n > 0$ .

For example,

- `get_n_strings(1)` returns `[""]`.
- `get_n_strings(4)` returns `["", "0", "01", "012"]`.
- `get_n_strings(7)` returns `["", "0", "01", "012", "0123", "01234", "012345"]`.

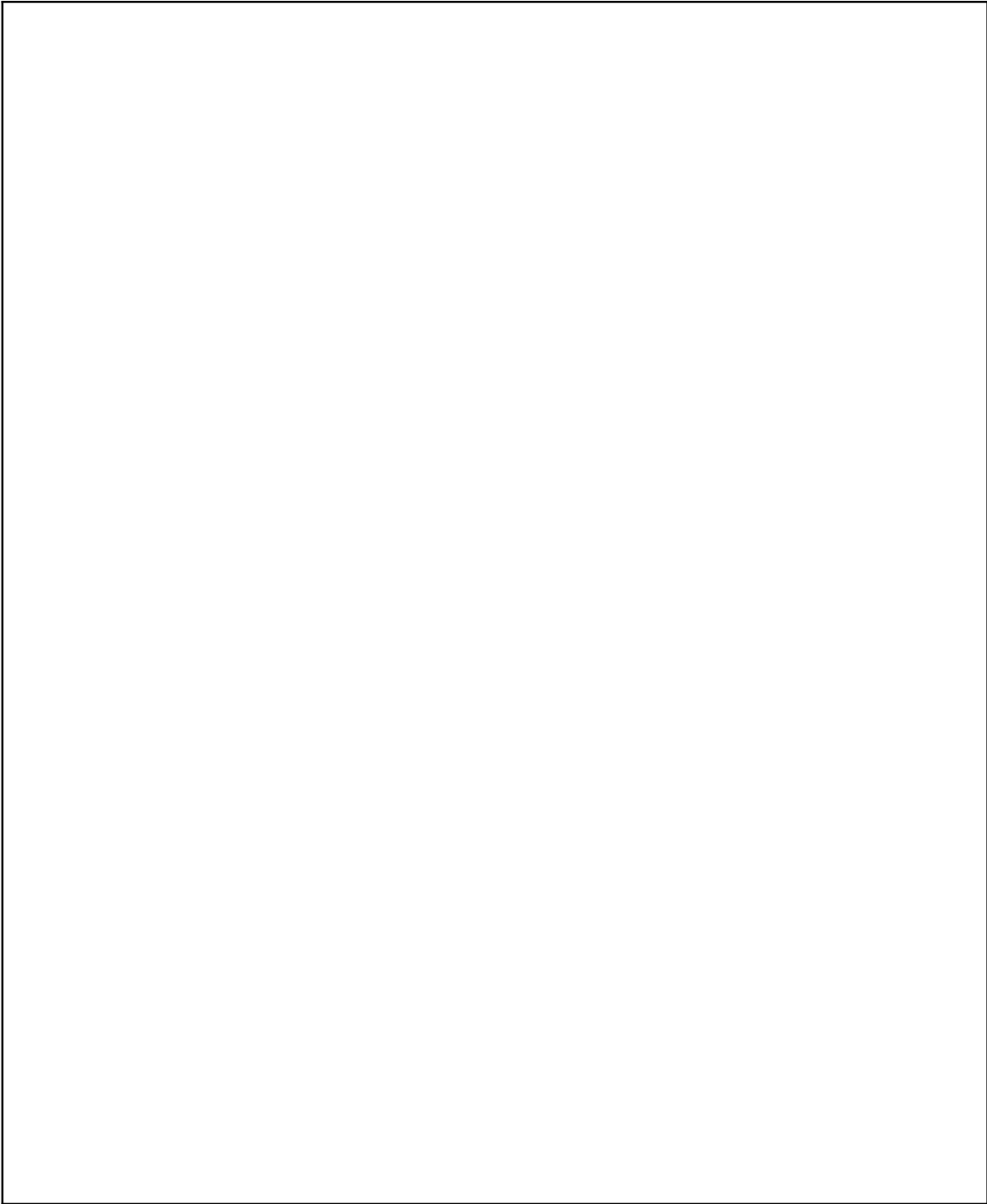
Make sure the returned structure is allocated on the heap.

You may use the standard functions if necessary (e.g., use the library `string.h`).

```
char** get_n_strings(int n) {
```

```
}
```

**Extra page**





**Empty page**