CMPT 125 D100, Fall 2025

Midterm Exam
October 28, 2025

Name_____

SFU ID: |__|__|__|__|__|__|__|__|__|

Problem 1	
Problem 2	
Problem 3	
Problem 4	
TOTAL	

Instructions:

1. Duration of the exam is 100 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 four (4) 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 [25 points]

a) [2 points each] Suppose you have a program in C named *program.c*.

1. What is the command in Linux to compile it into an executable file?

2. What is the default name of the compiled executable?

3. How can you specify the name of the executable in the compilation line?

b) [7 points] Will the code below compile? If yes, what will be the result of the execution? If there are any errors/warnings/potential issues explain them.

```
#include <stdio.h>
enum colors {RED, GREEN, BLUE, YELLOW};

int fun(int* x, int* y) {
      long z = 4;
      *y = z;
      *x = YELLOW;
      *y = 8;
      return z;
}

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;
}
```

c) [6 points] Will the code below compile? If yes, what will be the result of the execution? If there are any errors/warnings/potential issues explain them.

```
#include <stdio.h>

int main() {
    char str[15] = {0,28,29,30,31,'5','6','7',0};
    int ind=1;
    while (str[ind])
      ind++;

    printf("%s\n", str+ind-2);
    return 0;
}
```

d) [6 points] Will the code below compile? If yes, what will be the result of the execution? If there are any errors/warnings/potential issues explain them.

```
#include <stdio.h>

int* what() {
  int arr[3];
  for(int i=0;i<3;i++)
    arr[i] = i;
  return arr;
}

int main() {
  int* a = what();
  a[0] = 100;
  printf("a = [%d, %d, %d]\n", a[0], a[1], a[2]);
  return 0;
}
```

Problem 2 [25 points]

a) [15 points] Write a function that gets a string `str`, and a char `delim`, and returns the number of tokens in the string separated by `delim`.

For example,

- `count_tokens("abc-EFG--", '-')` *needs to return 2.*
- `count_tokens("++a+b+c+++", '+')` *needs to return 3.*
- `count_tokens("***", '*')` *needs to return 0.*
- `count_tokens("abcaa", '*')` *needs to return 1.*

```
int count_tokens(const char* str, char delim) {









}
```

b) [10 points] Write a function that gets two strings `str1` and `str2`, and checks if `str2` is a suffix `str1`. For full marks, your function needs to run in time O(strlen(str1)+strlen(str2)). For example,

- `is_suffix("abcd", "cd")` *needs to return* `true`.
- `is_suffix("cd", "Acd")` *needs to return* `false`.
- `is_suffix("123v787", "v787")` *needs to return* `true`.
- `is_suffix("xyz", "xyz")` *needs to return* `true`.

You may assume the standard libraries are included (e.g., *stdio.h*, *stdlib.h*, stdbool, *string.h*).

```
bool is_suffix(const char* str1, const char* str2) {




}
```

Problem 3 [25 points]

a) [5 points] Give an example of an array of length 10, on which **InsertionSort** makes exactly four swaps in each of the last two iterations of the outer loop, and no other swaps.

b) [5 points] Write a function called `pancake_flip`, that gets an array `arr` and integer `k`, and reverses the order of the first `k` elements in `arr` . We call it a `k-flip`.
For example,

- If `arr=[1,2,3,4,5,6,7,8]` and `k=3`, then after executing the function `arr` becomes `[3,2,1,4,5,6,7,8]`.
- If `arr=[1,2,3,4,5,6,7,8]` and `k=6`, then after executing the function `arr` becomes `[6,5,4,3,2,1,7,8]`.

(Note that k=0 or k=1 do not affect `arr`.) You may assume that the length of `arr` is at least `k`.

```
void pancake_flip(int* arr, int k) {




}
```

c) [15 points] Write a function `pancake_sort` that gets an array `A` and sorts it using a sequence of k-flips from the previous item. The function returns an array that represents the sequence of k-flips that sort `A`. For example,

- `pancake_sort(A =[3,2,4,1])` *can return the sequence* ***[5, 3,4,2,3,2]****.*
  The zeroth element is the length of the sequence (in this case 5), followed by a sequence of k's.
  By applying the 5 flips the result will be:
  3: `A =[4,2,3,1]`
  4: `A =[1,3,2,4]`
  2: `A =[3,1,2,4]`
  3: `A =[2,1,3,4]`
  2: `A =[1,2,3,4]`

Note: that the answer is not unique. For example for `A =[3,2,4,1])` *the function can also return the sequence* ***[4, 4,2,4,3]*** *-- 4 is the length of the sequence, and the flips are [4,2,4,3].*
Any solution with < 10*len(A) flips will be accepted for full marks.
Explain your idea before writing code!

```
// The function gets an array A of length len.
// It returns an array SEQ specifying the sequence of k-flips,
// so that the following code sorts A:
// ----------------------
//     for (i=1; i <= SEQ[0]; i++)
//       pancake_flip(A,SEQ[i]);
// ----------------------
// That is, SEQ[0] holds the length of the sequence,
// and A is sorted by the sequence of flips SEQ[1…SEQ[0]].
// (Note: the returned array must be allocated on the heap)
int* pancake_sort(int* A, int len) {









}
```

Problem 4 [25 points]

Consider the following function.

```
int foo(unsigned int n) {
  if (n<=3)
    return n;
  return 3*foo(n-1) + 2*foo(n-2) + 1*foo(n-3);
}
```

a) [5 points] Compute foo(5). Explain your answer.

b) [8 points] Rewrite the function foo() with the same functionality so that on input n, it returns the answer in time O(n). Explain your answer.

```
int foo(unsigned int n) {
```

c) [12 points] Consider the following definition of a Linked List.

```
struct LL_node {
  int data;
  struct LL_node* next;
};
typedef struct LL_node LL_node_t;

typedef struct {
  LL_node_t* head;
} LL_t;
```

Write a function that gets a Linked List and prints it in reverse order in time O(n), where n is the length of the list.

```
void print_reverse(LL_t* list) {
```

Extra page

Empty page