

CMPT125, Fall 2019

Homework Assignment 2

Due date: Wednesday, October 16, 2019, 23:59

You need to implement the functions in **assignment2.c**.
Submit only the **assignment2.c** file to CourSys.

Solve all 4 problems in the assignment.

The assignment will be graded both **automatically** and by **reading your code**.

Correctness: Make sure that your code compiles without warnings/errors, and returns the required output.

Readability: Your code should be readable. Add comments wherever is necessary. If needed, write helper functions to break the code into small, readable chunks.

Compilation: Your code MUST compile in CSIL with the Makefile provided. If the code does not compile in CSIL the grade on the assignment is 0 (zero). Even if you can't solve a problem, make sure it compiles

Warnings: Warnings during compilation will reduce points. More importantly, they indicate that something is probably wrong with the code.

Testing: An example of a test file is included. Your code will be tested using the provided tests as well as additional tests.

Question 1 [15 points].

Consider the following recursive function on positive integers:

```
foo(0) = 0
foo(1) = 1
foo(2) = 2
foo(n) = f(n-1) - 2f(n-2) + f(n-3) - 2 for n > 2
```

Write a function that gets an integer n and computes $foo(n)$.

```
long foo(int n)
```

Your function should return the correct answer in reasonable time on inputs up to 100.

Question 2 [20 points (10 points each item)]

A. Implement the recursive version of the linear search algorithm.

The function gets an (unsorted) array of length n and returns the index of item in the array.

If A does not contain the item, the function returns -1.

```
int linear_search_rec(int* A, int n, int item)
```

B. Implement the recursive version of the binary search algorithm.

The function gets a sorted array of length n and returns the index of item in the array.

If A does not contain the item, the function returns -1.

```
int binary_search_rec(int* A, int n, int item)
```

Question 3 [30 points].

Write a function that takes two positive integers k and n , and works as follows

1. prints all increasing sequences of length k consisting of the numbers $1 \dots n$
2. returns the number of such sequences. For example

```
int print_k_sequences(int n, int k)
```

For example, `print_k_sequences(5, 3)` prints the following sequences

```
1 2 3
1 2 4
1 2 5
1 3 4
1 3 5
1 4 5
2 3 4
2 3 5
2 4 5
3 4 5
```

and returns 10

The specific order in which the sequences are printed is up to you. Add exactly one space between numbers. Don't forget to separate the sequences with new lines.

Your function should work in reasonable time on inputs n, k up to 20.

[Hint: use recursion. You may also want to use a helper function]

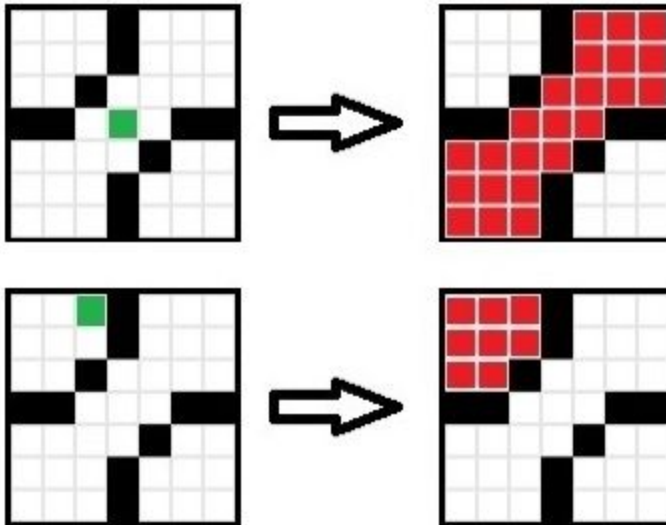
Question 4 [35 points].

Write a function that gets an NxN matrix of colors, and a starting point, and colors the WHITE area connected to the starting point with RED.

See examples below:

The *green point* on the left represents the starting point.

The *red points* are the area containing the starting point.



Specifically, the colors are represented as ints:

WHITE corresponds to the number 0, RED corresponds to 1.

All other numbers are treated as other colors.

The arguments of the function are the dimensions of the array and the starting point (see h. file for the definition of the type `point`)

```
void flood_fill(int N, int ar[N][N], point start)
```

The function colors the array as in flood fill algorithm.

Your function should work in reasonable time on arrays of size up to 50x50.

[Hint: use recursion]

[Hint2: the code should not be longer than 15 lines]