UCSC Silicon Valley Extension

C Programming 2 Test 1

- 1. Write a structure to store the following information for students in a school:
- (a) first name as a string with MAX characters
- (b) last name as a string with MAX characters
- (c) dorm name as a string with MAX characters
- (d) room number as an int

MAX has the value 50

- (a) Write a C program to read data for 10 students from the given file studentData.txt and store it in an array of the above structure.
- (b) Write a function called getNumStudentsInDorm that takes the name of a dorm as argument and displays the number of students that dorm.
- (c) Write a function called get getNumStudentsInRoom that displays the number of students in a given room number in all dorms.
- (d) Write a function called displayStudents that prints out the first name, last name, dorm name, and room number of all students in a dorm.

The file studentData.txt has information stored as follows:

10 # number of students

charles # student1 first name troy # student1 last name

NorthHouse # dorm name 1010 # room number

anna # student2 first name wang # student2 last name SouthHouse # d-

SouthHouse # dorm name 1110 # room number 2. Find the running time of the following functions using the asymptotic notation (O, o, w, Ω , Θ). Show the steps used to find the solution.

```
(a)
int func1(int n) {
     int val = 500;
     for (int i = 0; i < n; ++i) {
            val += i * 100;
     }
     return val;
}
(b)
int func2(int n) {
     int val1 = 0;
     for (int i = 0; i < n; ++i) {
         if (n < 50)
             val1 += 10;
         else
             val1 = func1(n);
(C)
 int val2 = 100;
 for (int i = 10; i < n * 10; ++i) {
      val2 = val2 /n;
      for (int j = 0; j < i; ++j) {
            val2 *= j + i;
      }
 }
```

3 Find the recurrence relation for the following programs. The recurrence does not have to be solved:

(a)

```
int func1(int n) {
    if (n < 1)
        return 1;
    else
        return func1(n-1) + func1(n-1) + func(n-1);
}

(b)
int func2(int N) {
        for (int i = 0; i < N/2; i+=1)
            sum += i;
        return 2*func2(N/2);
}</pre>
```

- 4. A 2D grid has a starting square S and a destination square D. The remaining squares have values from 0 to 2. Write a program that finds the maximum value of any path from S to D with the following rules:
- 2. When a square with a non-zero value is exited, its value decreases by 1. For example, when a square with value 2 is exited, its value changes to 1. When a square with value 1 is exited, its value changes to 0.
- 3. Squares whose values are zero remain unchanged when the square is entered or exited.
- 3. A value of 0 is returned if there is no path from *S* to *D*.

Example 1: Path 1 goes through one square with value 1. The maximum value of this path is 1.

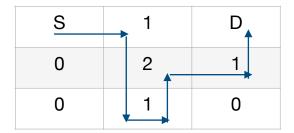
| S | 1 | D |
|---|---|---|
| 0 | 2 | 1 |
| 0 | 1 | 0 |

After the square at (0,1) is exited, its value changes from 1 to 0 as shown below:

| S | 0 | D |
|---|---|---|
| 0 | 2 | 1 |

| 0 | 1 | 0 |
|---|---|---|
| | | |

Example 2: Path 2 goes through squares (0,1), (1, 1), (2,1), (1,1), and (1,2). The maximum value of this path is 6. Square (1,1) is exited twice with values 2 and 1 and the other three squares (0,1), (2,1) and (1,2) with value 1 are exited once.



No other path has a higher value and so the maximum value from S to D is 6, which is the solution.

The test case file has 5 test cases and all arrays have 5 rows and 5 columns.

S000D

11111

00000

22222

00000

S111D

02221

01111

00000

00000

00000

S0000

00000

11111

22222

0000D

S1111

10221

22212

00001

0000D

S1111

10221 22212

00001

0000D

For problem 1, studentData.txt is provided below:

10

anna

wang

NorthHouse

1010

rob

keller

SouthHouse

1110

joanne

mana

SouthHouse

1110

chase

steele

EastHouse

1050

herb

phillips

SouthHouse

1110

buck

davidson

SouthHouse

1120

vilma

wolfe

EastHouse

1110

felice

odon

SouthHouse

1110

john

camacho

SouthHouse

1110

tracey

haye

SouthHouse 1050