CS 137: Assignment #4

Due on Friday, Oct 11, 2024, at 11:59 PM

Submit all programs using the Marmoset Submission and Testing Server located at
https://marmoset.student.cs.uwaterloo.ca/

Victoria Sakhnini

Fall 2024

Notes:

- Use the examples to guide your formatting for your output. Remember to terminate your output with a newline character.
- Integers should be read using `scanf`.
- For this and all future assignments, I strongly recommend that you solve the extra practice problems in the course notes before working on the assignment.
- For this assignment, you may use any content covered until the end of chapter 7.
- **You must create more tests. Examples don't cover all the cases.**
- **You may use MATH Library for only problems 3 and 4.**
- **You must NOT use a variable-length array.**
- **You are free to decide whether to use loops or recursion to solve any of the questions.**

Problem 1

Write the function `int removekdigits(long long int n, int k);` that returns an integer like n after removing the first k largest digits from n, and returns the smallest integer possible from the resulting digits. Return 0 if no digits remain after removing k digits. Assume k<= length of n.

- **You must NOT use a variable length array (VLA). Using VLA will result in receiving a zero on this question.**
- **Assume n>=0**

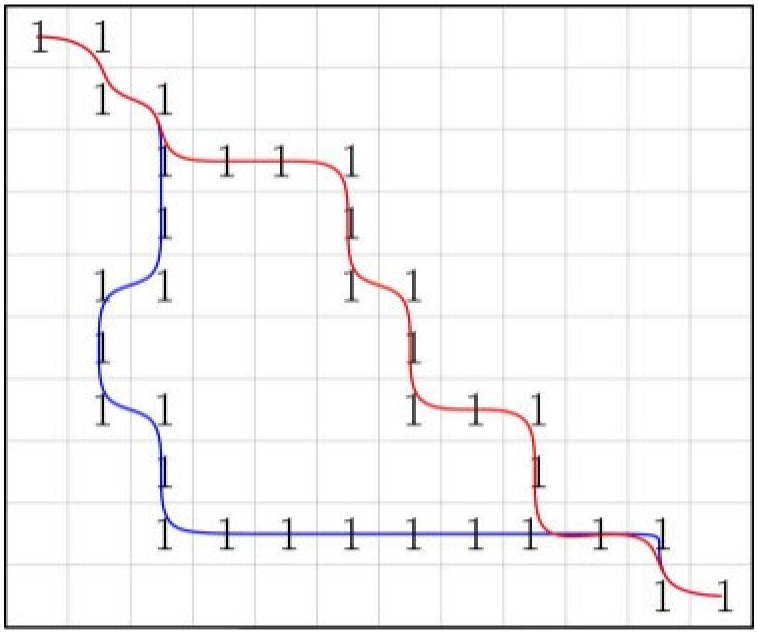
Submit `kdigits.c` file containing only your implemented functions (that is, you must delete the test cases portion/the `main` function). However, **you must keep the required included libraries.**

Sample Test program (make sure you test your solution with more test cases)

```
int main(void) {

    assert(removekdigits(345267,3)==234);
    assert(removekdigits(0,0)==0);
    assert(removekdigits(9,1)==0);
    assert(removekdigits(871,0)==178);
    assert(removekdigits(9898989,2)==88899);

    return 0;
}

```

Problem 2

Let *a* be an mxn matrix with 0 and 1 entries. Assume the top left entry `a[0][0] = 1` and the bottom right entry `a[m-1][n-1] = 1`.
A ***path*** between the top left and bottom right entries is a sequence of matrix entries such that adjacent entries in the path are adjacent in the matrix.
The figure below gives two such paths (assume the blank entries are zeros in the matrix).



A **monotone path** is a path that does not travel left or up. In the figure, the red path is monotone. The blue path is not, because it goes left.

Write a function `int monotonePath(int m, int n, int a[m][n])` that returns 1 if the matrix `a` has a monotone path from `a[0][0]` to `a[m-1][n-1]`, otherwise return 0.

Submit `path.c` file containing only your implemented function (that is, you must delete the test cases portion/the main function). However, **you must keep the required included libraries.**

Sample Test program (You need to add more test cases)

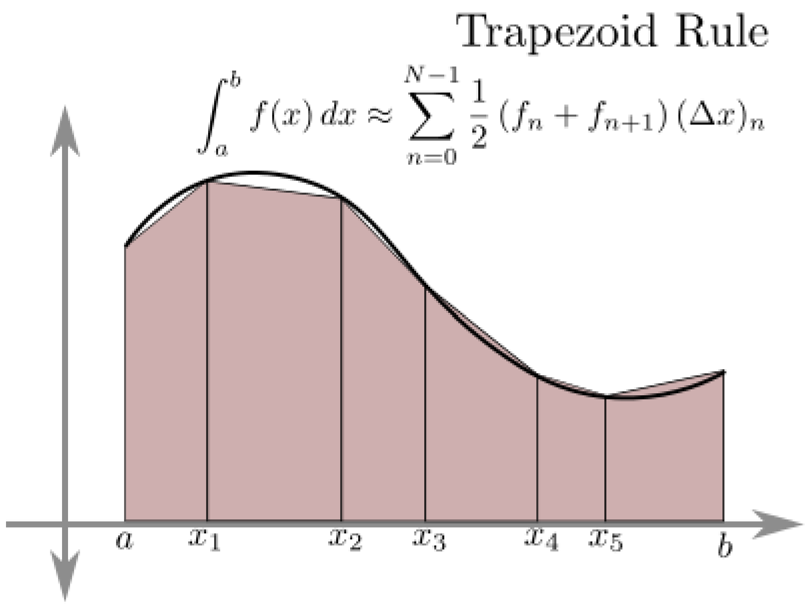
```
int main(void)
{
    int m = 10;
    int n = 12;
    int a[][12] = {{1,1,0,0,0,0,0,0,0,0,0,0},
                   {0,1,1,0,0,0,0,0,0,0,0,0},
                   {0,0,1,1,1,1,0,0,0,0,0,0},
                   {0,0,1,0,0,1,0,0,0,0,0,0},
                   {0,1,1,0,0,1,1,0,0,0,0,0},
                   {0,1,0,0,0,0,1,0,0,0,0,0},
                   {0,1,1,0,0,0,1,1,1,0,0,0},
                   {0,0,1,0,0,0,0,0,1,0,0,0},
                   {0,0,1,1,1,1,1,1,1,1,1,0},
                   {0,0,0,0,0,0,0,0,0,0,1,1}
    };

    // a, but with one entry switched eliminating the monotone path
    int b[10][12] = {{1,1,0,0,0,0,0,0,0,0,0,0},
                     {0,1,1,0,0,0,0,0,0,0,0,0},
                     {0,0,1,1,0,1,0,0,0,0,0,0},
                     {0,0,1,0,0,1,0,0,0,0,0,0},
                     {0,1,1,0,0,1,1,0,0,0,0,0},
                     {0,1,0,0,0,0,1,0,0,0,0,0},
                     {0,1,1,0,0,0,1,1,1,0,0,0},
                     {0,0,1,0,0,0,0,0,1,0,0,0},
                     {0,0,1,1,1,1,1,1,1,1,1,0},
                     {0,0,0,0,0,0,0,0,0,0,1,1}
    };

    assert (monotonePath(m, n, a));
    assert (!monotonePath(m, n, b));
    return 0;
}

```

Problem 3

Complete the provided C program named `trapezoidal.c` that consists of the function

```
// b>a, n>0

double trapezoidal(double (*f)(double), double a, double b, double epsilon, int
n);
```

that computes an approximation to find the definite integral of a function.


Trapezoid Rule
$\int_a^b f(x)\,dx \approx \sum_{n=0}^{N-1} \frac{1}{2}(f_n + f_{n+1})(\Delta x)_n$
a
x_1
x_2
x_3
x_4
x_5
b

The function f(x) is divided into many sub-intervals, and each interval is approximated by a Trapezium. Then the area of trapeziums is calculated to find the integral, which is the area under the curve. The more the number of trapeziums is used, the better the approximation.

Trapezodial begins by taking the dividing the area into n sub-intervals, calculate the area of trapeziums, then continue doing the same by doubling the number of sub-intervals until the difference between the last two results (area of n trapeziums vs area of 2n trapeziums) calculated is less or equal epsilon.

Note: You are to submit this file containing only your implemented function (that is, you must delete the test cases portion and the `main` function). However, **you must keep the required included libraries.**

The sample code for testing is below.

```c
#include <stdio.h>
#include <math.h>
#include <assert.h>
#define PI acos(-1)

double f1(double x)
{
         return x * x;
}

 double f2(double x)
 {
         return cos(x);
 }

 double f3(double x)
 {
         return sqrt(x);
 }

 double trapezoidal(double (*f) (double), double a, double b, double epsilon, int n)
 {

 }

 int main(void)
 {
  assert(fabs(trapezoidal(f1, 5, 10, 0.00001, 2) - 291.667) <= 0.001);
  assert(fabs(trapezoidal(f2, PI/2, 3, 0.00001, 5) - -0.858879) <=
         0.000001);
 assert(fabs(trapezoidal(f3, 1, 4, 0.00001, 2) - 4.666) <= 0.001);
 return 0;
 }

```

Problem 4

Write a C program `grades.c` that reads the grades of a certain course and provides some statistics as illustrated in the examples below. Please note the following:

- Your program must read grades using `scanf` until a character is entered.
- Assume that all entered grades are between 0 – 100, inclusive. (This is a critical point to be able to solve the problem without using VLA)
- We also provided `string.txt` to avoid any typos in formatting the outputs. **Make sure to use it for printing**.
- Assume that the user will enter at least two valid grades.
- ~~Assume that at least one grade is entered.~~
- Failing grade is a grade <50.
- **You must NOT use a variable length array (VLA). Using VLA will result in receiving a zero on this question.**
- **You may not limit the maximum number of grades that can be entered. Failing to do so will result in receiving a zero on this question.**

The **sample standard deviation** formula looks like this:

Formula	**Explanation**
$s = \sqrt{\frac{\sum (X - \bar{x})^2}{n-1}}$	• s = sample standard deviation • $\sum$ = sum of... • X = each value • $\bar{x}$ = sample mean • n = number of values in the sample

$$\text{mean} = \overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Sample input 1:

```
80
78
80
96
85
78
85
85
9
44
45
66
45
13
?
```

Sample output 1:

```
The maximal grade is: 96
The number of students who received the maximal grade is 1
The minimal grade is: 9
The number of students who received the minimal grade is 1
The course mean is: 63.500
The standard deviation is: 27.734
The number of students who failed the course is: 5
```

Sample input 2:

```
100
100
100
50
30
30
30
30
0
0
0
*
```

Sample output 2:

```
The maximal grade is: 100
The number of students who received the maximal grade is 3
The minimal grade is: 0
The number of students who received the minimal grade is 3
The course mean is: 42.727
The standard deviation is: 40.023
The number of students who failed the course is: 7
```