

**Week 1: 7-Dec-2021 (Scan, print, arithmetic opr.)**

**w01-1.** Write a C program that prints your name and roll number in two consecutive lines.

The name of your C file should be w01-1.c.

**Example**

```
Name: Aritra Basu
Roll number: 17AG21015
```

**w01-2.** Write a C program that takes as input two integers  $a$  and  $b$  from the keyboard, computes  $a/b$ , and prints its value on the terminal.

- ◆ The value of  $a/b$  should be correct up to 6th decimal place.
- ◆ The name of your C file should be w01-2.c.

**Example**

```
Enter a and b: 2 3
a/b = 0.666667
```

**w01-3.** Write a program that takes as input the value of a floating-point variable, say  $x$ , calculates the value of the expression  $1 + x + x^2$ , and prints the result on the terminal correct up to  $10^{\text{th}}$  decimal place.

- ◆ Assume that the value of  $x$  is positive and less than 1.
- ◆ You cannot use math library.
- ◆ The name of your C file should be w01-3.c.

**Examples**

```
Enter x: .5
Answer = 1.7500000000
```

```
Enter x: .31
Answer = 1.4061000000
```

```
Enter x: .9999
Answer = 2.9997000100
```

## Week 2: 14-Dec-2021 (Data types, variables and constants, operators, basic I/O, assignment expressions.)

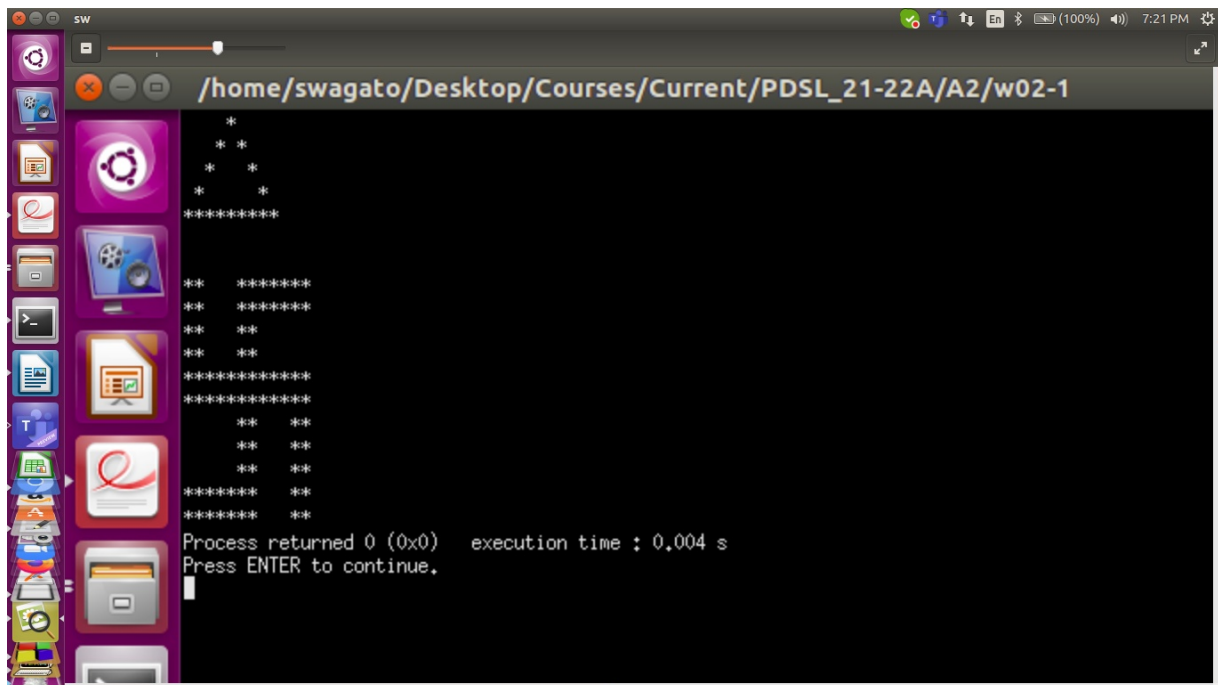
### Instructions:

- Use of advanced concepts like conditionals (if, if-else, switch, etc.), loops, functions, array, etc. is not permitted and will attract penalty.
- Use of no library function other than printf and scanf is permitted.
- Your programs should produce the sample outputs exactly.

**w02-1.** Write a C program that prints the following two patterns (a triangle and a Swastika) on the console. Your printed patterns should be exactly as the sample output. However, your number of newlines between the triangle and Swastika may differ from ours. The name of your C file should be w02-1.c.

[5 + 5]

### Sample output



```
SW
/home/swagato/Desktop/Courses/Current/PDSL_21-22A/A2/w02-1
*
* *
* * *
* * *
*****
** *****
** *****
** *****
** *****
*****
*****
** **
** **
** **
***** **
***** **
Process returned 0 (0x0)   execution time : 0.004 s
Press ENTER to continue.
```

**w02-2.** Write a program that accomplishes the following tasks. You should submit a single C file named w02-2.c. Your program should

- ◆ take in a floating point number  $a$  ( $0 < a < 1$ ) as input through the keyboard,
- ◆ print  $a$  on the console correct up to **exactly** 4 places after the decimal point, [1]
- ◆ evaluate the polynomial  $p(x) = 1 + x + x^2 + \dots + x^{15}$  at  $x = a$ , and print the evaluation  $p(a)$  on the console in a new line correct up to **exactly** 4 decimal places, [3]
- ◆ evaluate the polynomial  $q(x) = 1 + 2x + 3x^2 + \dots + 16x^{15}$  at  $x = a$ , and print the evaluation  $q(a)$  on the console in a new line correct up to **exactly** 4 decimal places, [3]
- ◆ evaluate  $r(x) = \lim_{n \rightarrow \infty} 1 + x + x^2 + \dots + x^n$  at  $x = a$ , and print the evaluation  $r(a)$  on the console in a new line correct up to **exactly** 4 decimal places. [3]

Marks will depend on the efficiency of your code. Try to use as few arithmetic operations (addition, subtraction, multiplication and division) as possible.

### Sample inputs and outputs:

(a)

```
Enter a: 0.43
a=0.4300
p(a)=1.7544
q(a)=3.0778
r(a)=1.7544
```

(b)

```
Enter a: 0.56
a=0.5600
p(a)=2.2725
q(a)=5.1614
r(a)=2.2727
```

- w02-3.** User supplies a positive integer having value at most 9999. Your program should shift its digits cyclically towards right, one digit at a time, and print the results on the terminal. This should continue 4 times. The program should treat the input as a 4-digit number, by adding 0 to its left if needed. The numbers printed should be right-justified. You should use as few variables as possible. Your marks will depend on the number of variables you use. You should submit a single C file named w02-3.c.

[10]

### Sample inputs and outputs:

(a)

```
Enter a positive integer: 3251
1325
5132
2513
```

3251

(b)

Enter a positive integer: 23

3002

2300

230

23

(c)

Enter a positive integer: 1313

3131

1313

3131

1313

## Week 3: 21-Dec-2021 (Logical expression, conditionals (if-else, switch-case))

**w03-1.** Write a C program that does the following:

- ◆ The program first takes in a 7- or 8-digit positive integer as input (that's ensured) through the keyboard that is to be interpreted as a date in *dd-mm-yyyy* format.  
Example: 22112001 stands for November 22, 2001 and 3011987 stands for January 3, 1987. Note that the last 4 digits correspond to year, the next 2 digits correspond to month, and the next 1 or 2 digits correspond to day. Make sure that you use an integer type that can accommodate any 8-digit integer. You may use the *sizeof( )* function to find out the sizes of various data types in your system.
- ◆ Prints the numbers in the day, month, and year fields, each in a new line. [2]
- ◆ If the year is before 2021 or after 2099, then the program prints an appropriate message and terminates. [1]
- ◆ If the input integer does not correspond to any valid date (for example, 99092022, 31062078, 29022023), then the program prints an explanatory error message and terminates. If there are multiple explanations of invalidity (for example, both day and month are out of their respective ranges), any one explanation can be printed. [2]
- ◆ If the input integer corresponds to a valid date, and the year is between 2021 and 2099 (both inclusive) then the program prints the month and the day (both in words) in a new line. [1+4]

You may use the fact that January 1, 2021 is a Friday.

The name of your C file should be w03-1.c.

### Sample inputs and outputs:

```
Enter date: 28768099
Day: 28
Month: 76
Year: 8099
After 2099
```

```
Enter date: 30052010
Day: 30
Month: 5
Year: 2010
Before 2021
```

```
Enter date: 32162025
Day: 32
Month=16
Year=2025
Month out of range.
```

```
Enter date: 29022023
Day: 29
Month: 2
Year: 2023
Day out of range.
```

```
Enter date: 1012064
Day: 1
Month: 1
Year: 2064
It's a Tuesday of January.
```

```
Enter date: 22122021
Day: 21
Month: 12
Year: 2021
It's a Wednesday of December.
```

**w03-2.** Write a program that manages transactions in a snacks bar. The program should do the following:

- ◆ The program first displays a header line:  
Following are 12 items and their rates (INR):  
Then it skips a line and prints a menu on the console, as shown below. [3]
- ◆ Next, the program skips a line and prints:  
Enter the Serial Number (SN) of item and the quantity you want:  
The user will enter two valid integers in response, the first being the SN and the second its quantity. [1]
- ◆ After the user's input, the program displays the item name and the quantity requested. After that, it computes the payable amount and displays the calculation. The basic price is the rate of the item times the quantity. The tax is 12.5% of the basic price. The payable amount is the sum of the basic price and the tax, rounded to its nearest integer (e.g., 10.49 is rounded to 10, while 10.50 or 10.60 is rounded to 11). Finally, the program prints a greeting. [6]

Everything printed should exactly match the output given below, including alignments, justifications, precision of the fractions, the horizontal lines, the greeting, etc. You may assume that the total amount (before rounding) is less than INR 9999.50.

The name of your file should be w03-2.c.

### Sample input and output

Following are 12 items and their rates (INR):

SN	ITEM	RATE
--	----	----
01:	BUTTER COOKIES	25
02:	CASHEW COOKIES	30
03:	CREAM CAKE	22
04:	LEMON JUICE	35
05:	VEG CASHEW CAKE	18
06:	MANGO JUICE	78
07:	COOKIES (PLAIN)	15
08:	ORANGE JUICE	73
09:	MILK BISCUITS	12
10:	PLAIN VEG CAKE	20
11:	BUTTER FRUIT CAKE	25
12:	PINEAPPLE JUICE	65

```
Enter the Serial Number (SN) of item and the quantity you want: 8 7
Your item is ORANGE JUICE x 7
Basic price = Rs. 511.00
Tax @12.5% = Rs. 63.88
Total price = Rs. 574.88
=====
To pay      = Rs. 575.00
=====
Thank you!
Please visit us again.
```

**w03-3.** Any fraction  $\frac{a}{b}$  can be expressed as a continued fraction as follows:

$$p_1 + \frac{1}{p_2 + \frac{1}{p_3 + \frac{1}{p_4 + \frac{1}{\ddots}}}}$$

For convenience, it is represented as a sequence  $[p_1; p_2, p_3, p_4, \dots]$ , where  $p_1 \geq 0$  and  $p_i \geq 1$  if  $i \geq 2$ .

**Examples:**

(a)  $\frac{21}{7} = [3]$

(b)  $\frac{22}{7} = [3; 7]$

(c)  $\frac{333}{106} = 3 + \frac{1}{7 + \frac{1}{15}} = [3; 7, 15]$

(d)  $\frac{104348}{33215} = 3 + \frac{1}{7 + \frac{1}{15 + \frac{1}{1 + \frac{1}{293}}}} = [3; 7, 15, 1, 293]$

Given a numerator  $a$  and a denominator  $b$  as input, your program has to print the sequence, as shown in the examples. You can use one extra integer variable, say  $c$ ; use of more extra variables will attract penalty. You should not use math library or any function other than printf and scanf.

Assume that the input is such that there are at most five terms in the sequence. In the above examples, the first fraction has just one term and the last one has five terms.

The name of your file should be w03-3.c. **[10]**

**Sample input and output**

```
Enter a and b: 21 7
[3]
```

```
Enter a and b: 22 7
[3; 7]
```

```
Enter a and b: 333 106
[3; 7, 15]
```

```
Enter a and b: 104348 33215
[3; 7, 15, 1, 293]
```

## Week 4: 28-Dec-2021 (Loops)

### Instructions:

- Use of advanced concepts like functions, array, pointers, etc. is not permitted.
- Use of library functions other than printf and scanf is not permitted.

**w04-1.** This is a continuation and generalization of w03-3. Given a numerator  $a$  and a denominator  $b$  as input, your program has to print the sequence representing the continued fraction for  $a/b$ , as shown in the examples. You can use one extra integer variable, say  $c$ ; use of more extra variables will attract penalty.

You should use a loop because the sequence may be arbitrarily long.

### Sample input and output

```
Enter a and b: 1 2
Continued fraction = [0; 2].
```

```
Enter a and b: 2 1
Continued fraction = [2].
```

```
Enter a and b: 1 1
Continued fraction = [1].
```

```
Enter a and b: 2 2
Continued fraction = [1].
```

```
Enter a and b: 3 6
Continued fraction = [0; 2].
```

```
Enter a and b: 6 3
Continued fraction = [2].
```

```
Enter a and b: 333 106
[3; 7, 15]
```

```
Enter a and b: 104348 33215
[3; 7, 15, 1, 293]
```

```
Enter a and b: 12213579 213
Continued fraction = [57340; 1, 2, 1, 17].
```

```
Enter a and b: 13578642 8642135
Continued fraction = [1; 1, 1, 3, 94, 1, 2, 1, 43, 1, 2, 1, 5, 3].
```



**w04-2.** In this problem you will find an approximate root of a cubic polynomial, by a method described below. It is mandatory to stick to this method.

- (1) The program accepts the coefficients of a univariate cubic polynomial as input from the user. Assume that the coefficients are all integers in the range  $[-5, 5]$ . Assume further that the leading coefficient (coefficient of  $x^3$ ) is non-zero. Call the polynomial  $p()$ .
- (2) Define two variables of type *double*, named  $a$  and  $b$ . Initialize them so that  $a < b$ , and  $p(a)$  and  $p(b)$  are of opposite signs (that is one of them is positive and the other is negative). Note that the interval  $[a, b]$  contains a root of  $p()$ . The idea is to shrink the interval over iterations and converge to the root.
- (3) Repeat:
  - i. Let  $(c, 0)$  be the point where the straight line segment joining points  $(a, p(a))$  and  $(b, p(b))$  intersects the X-axis. Find  $c$  and compute  $p(c)$ .
  - ii. If the absolute value of  $p(c)$  is less than 0.001, then print  $c$  as a root and terminate.
  - iii. If the signs of  $p(a)$  and  $p(c)$  are the same, then update  $a$  to  $c$ . Otherwise update  $b$  to  $c$ .

You should consider all real-domain computations in *double precision*.

**Sample input and output:**

```
Enter coefficient of x^0: 3
Enter coefficient of x^1: -2
Enter coefficient of x^2: 4
Enter coefficient of x^3: 1
Root of the polynomial: -4.579702
```

**w04-3.** An *integer point* means its  $(x, y)$  coordinates are integers. Given as the input the coordinates of two distinct integer points,  $p$  and  $q$ , your program has to find the number of integer points on the straight line segment  $pq$  and has to print the coordinates of these points. You should use only integer variables and integer computations in your program. [5+5]

**Sample input and output:**

```
Enter (x,y) coordinates of Point p: 0 0
Enter (x,y) coordinates of Point q: 1 2
Number of integer points on the line segment pq = 2.
Integer points on the line segment pq:
1: (0,0)
2: (1,2)

Enter (x,y) coordinates of Point p: -2 -3
Enter (x,y) coordinates of Point q: 2 3
Number of integer points on the line segment pq = 3.
Integer points on the line segment pq:
1: (-2,-3)
2: (0,0)
3: (2,3)

Enter (x,y) coordinates of Point p: -2 -2
Enter (x,y) coordinates of Point q: 3 3
Number of integer points on the line segment pq = 6.
```

Integer points on the line segment pq:

1: (-2,-2)  
2: (-1,-1)  
3: (0,0)  
4: (1,1)  
5: (2,2)  
6: (3,3)

Enter (x,y) coordinates of Point p: 0 0

Enter (x,y) coordinates of Point q: 15 19

Number of integer points on the line segment pq = 2.

Integer points on the line segment pq:

1: (0,0)  
2: (15,19)

Enter (x,y) coordinates of Point p: 0 0

Enter (x,y) coordinates of Point q: 15 20

Number of integer points on the line segment pq = 6.

Integer points on the line segment pq:

1: (0,0)  
2: (3,4)  
3: (6,8)  
4: (9,12)  
5: (12,16)  
6: (15,20)

Enter (x,y) coordinates of Point p: -10 10

Enter (x,y) coordinates of Point q: 50 52

Number of integer points on the line segment pq = 7.

Integer points on the line segment pq:

1: (-10,10)  
2: (0,17)  
3: (10,24)  
4: (20,31)  
5: (30,38)  
6: (40,45)  
7: (50,52)

Enter (x,y) coordinates of Point p: -137552 113409

Enter (x,y) coordinates of Point q: 271895 579

Number of integer points on the line segment pq = 2.

Integer points on the line segment pq:

1: (-137552,113409)  
2: (271895,579)

Enter (x,y) coordinates of Point p: -137552 113409

Enter (x,y) coordinates of Point q: -271895 -579

Number of integer points on the line segment pq = 4072.

Integer points on the line segment pq:

1: (-137552,113409)  
2: (-137585,113381)  
3: (-137618,113353)  
...  
100: (-140819,110637)

101: (-140852,110609)  
102: (-140885,110581)

...

500: (-154019,99437)  
501: (-154052,99409)  
502: (-154085,99381)

...

1000: (-170519,85437)  
1001: (-170552,85409)  
1002: (-170585,85381)

...

4070: (-271829,-523)  
4071: (-271862,-551)  
4072: (-271895,-579)

## Week 5: 4-Jan-2022 (Arrays 1D, simple examples of searching & sorting.)

### Instructions:

- Use of advanced concepts like functions, pointers, etc. is not permitted.
- Use of library functions other than printf and scanf is not permitted.

**w05-1.** In this problem you will write a program that maintains a sorted list of distinct integers, by using a one-dimensional array. The following should be the input/output behavior of your program. Your program should also perfectly match the sample inputs and outputs given below. ( $5 \times 2 = 10$ )

- ◆ The program should first output a menu listing five options numbered 1 through 5, and ask the user to enter one of those options. See sample output.
- ◆ If the user enters a number other than 1-5, the program prompts the user to enter a valid option number and displays the menu again.
- ◆ If the user enters 1, the program asks the user for an integer. If the integer does not exist in the list, it is now inserted into the list, and the currently there are less than 100 elements in the list. The list should be sorted after the insertion. If the integer already exists in the list or there are already 100 elements in the list before insertion, the program displays an appropriate message (see sample outputs). After that, the menu of options is displayed again.
- ◆ If the user enters 2, the program asks the user to enter a range  $[\ell, r]$ . The program should then delete all integers in the list that are between  $\ell$  and  $r$ , including both. At the end of deletion, the list should be sorted. If there is no integer in the list between  $\ell$  and  $r$  before deletion, a message should be displayed. Finally, the menu of options should be displayed again.
- ◆ If the user enters 3, the program asks for an input. The program then displays whether or not that integer exists in the list. This operation should not alter the list. Finally, the menu of options is displayed again.
- ◆ If the user enters 4, the program displays all the integers currently in the list, each in a new line. This operation should not alter the list. Note that the integers are displayed in sorted order. Finally, the the menu of options is displayed again.
- ◆ If the user enters 5, your program should terminate.

### Sample inputs and outputs:

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 4  
Database empty.

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 3  
Database empty.

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 1

Enter number: 34

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 1

Enter number: 56

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 1

Enter number: 41

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 4

Numbers in the database:

34

41

56

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 2

Enter lower limit: 5

Enter upper limit: 18

No number in the given range

1. Insert
2. Delete numbers in a range
3. Search

4. Display
5. Exit

Enter an option: 2  
Enter lower limit: 30  
Enter upper limit: 41

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 1  
Enter number: 56  
Number exists!

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 4  
Numbers in the database:  
56

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 2  
Enter lower limit: 58  
Enter upper limit: 60  
No number in the given range

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 2  
Enter lower limit: 56  
Enter upper limit: 100

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 4  
Database empty.

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 3  
Database empty.

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 1  
Enter number: 43

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 1  
Enter number: 34

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 1  
Enter number: 76

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 4  
Numbers in the database:  
34  
43  
76

1. Insert

2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 3

Enter number: 36

Number does not exist.

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 3

Enter number: 43

Number exists.

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 7

Enter a valid option

1. Insert
2. Delete numbers in a range
3. Search
4. Display
5. Exit

Enter an option: 5

**w05-2.** Write a program that

- ◆ First takes in four integers  $a, b, c$  and  $d$  as inputs. Assume that  $0 \leq a/b < c/d \leq 1$ .
- ◆ Take in an integer  $n$  as input. Assume that  $n \leq 100$ .
- ◆ Declare an array  $A$  of size as much as needed ( $= 2 \times (100 + 2) = 204$ ). In this array, the numerator and denominator of each fraction will be stored in two consecutive cells.
- ◆ Store  $a$  and  $b$  in  $A[0]$  and  $A[1]$  respectively. [ $c$  and  $d$  will be stored at the end.]
- ◆ Generate, print and store  $n$  distinct fractions in the open interval  $(a/b, c/d)$  in increasing order as follows:
  - ◆ Print the fraction  $(a + c)/(b + d)$ . Then, find its reduced form  $f_{11}/f_{12}$  and print it. After that, store  $f_{11}$  and  $f_{12}$  in next two successive cells of  $A$ .
  - ◆ Print the fraction  $(f_{11} + c)/(f_{12} + d)$ . Then, find its reduced form  $f_{21}/f_{22}$  and print it. After that, store  $f_{21}$  and  $f_{22}$  in next two successive cells (corresponding to indices 6 and 7) of  $A$ .
  - ◆ Continue this for  $n - 2$  more steps.
- ◆ Store  $c$  and  $d$  in the next two successive cells in  $A$ .
- ◆ Finally, print  $a/b$ , the  $n$  fractions stored in the last sequence of steps in that order, and  $c/d$  (in this order). Print each fraction in a new line.

The output of your program must exactly match the sample outputs given below. **(10)**



### Sample input and output:

Enter numerator and denominator of fraction 1 in reduced form: 1 5  
Enter numerator and denominator of fraction 2 in reduced form: 2 7  
Enter number of fractions: 10

Fraction 1: 3/12  
GCD of numerator and denominator=3  
Fraction 1 in reduced form =1/4

Fraction 2: 3/11  
GCD of numerator and denominator=1  
Fraction 2 in reduced form =3/11

Fraction 3: 5/18  
GCD of numerator and denominator=1  
Fraction 3 in reduced form =5/18

Fraction 4: 7/25  
GCD of numerator and denominator=1  
Fraction 4 in reduced form =7/25

Fraction 5: 9/32  
GCD of numerator and denominator=1  
Fraction 5 in reduced form =9/32

Fraction 6: 11/39  
GCD of numerator and denominator=1  
Fraction 6 in reduced form =11/39

Fraction 7: 13/46  
GCD of numerator and denominator=1  
Fraction 7 in reduced form =13/46

Fraction 8: 15/53  
GCD of numerator and denominator=1  
Fraction 8 in reduced form =15/53

Fraction 9: 17/60  
GCD of numerator and denominator=1  
Fraction 9 in reduced form =17/60

Fraction 10: 19/67  
GCD of numerator and denominator=1  
Fraction 10 in reduced form =19/67

All fractions:

1/5  
1/4  
3/11  
5/18  
7/25  
9/32

11/39  
13/46  
15/53  
17/60  
19/67  
2/7

## Week 6: 11-Jan-2022 (1D-array and function)

### Instructions:

- Use of advanced concepts like 2D-arrays, structures, pointers, dynamic memory allocation is not permitted and will attract penalty.
- Use of no library function other than `printf`, `scanf`, `rand`, and `srand` is permitted.

**w06-1.** The task is to generate  $n$  random points on the plane.  
For this, write a C program that has the following input/output behavior:

- ◆ The program asks the user for a positive integer  $n$ . Assume that  $n \leq 400$ .
- ◆ The program asks the user for a seed (a non-negative integer) for random number generation.
- ◆ The program generates  $n$  **distinct** points randomly on the 2D-plane, whose coordinates are integers in  $[0, 20]$ , and stores them in a 1D integer array  $A$ . The first two cells of  $A$  should contain the  $(x, y)$  coordinates of the first point, the second two cells for the second point, and so on. The program starts with the array  $A$  as empty. Each time it generates a point, it first searches  $A$  to see if the point already exists in  $A$ . (Write a function to do this.) If the point exists, then it is not inserted in  $A$  and a new point is generated. If the point does not exist in  $A$ , then it is inserted in  $A$ . It stops inserting points when the number of points in  $A$  becomes  $n$ .
- ◆ The program prints all the points in  $A$ . Each point is printed as a space-separated pair of integers in a new line (see sample output).

To generate the random numbers, use the `srand` and `rand` functions defined in `stdlib.h`.

Their prototypes are as follows.

```
void srand (unsigned s);  
int rand (void);
```

First, you have to set the seed to the user input (say,  $s$ ) as follows.

```
srand(s);
```

Then call `rand()` to get a random integer in the interval  $[0, \text{RAND\_MAX}]$ , and assign it to an integer variable, say  $r$ : `r=rand()`;

Note:

`RAND_MAX` is a large integer constant whose value need not concern us.

Then you can do `r = r%21;`

so as to get  $r$  in the interval  $[0, 20]$ .

**Sample input and output** (notice that with changing seed, you get changing points):

```
Enter number of points: 7  
Enter seed: 1  
Unique points generated:  
1 4  
9 19  
8 10  
10 9  
15 10  
2 19  
20 4
```

```
Enter number of points: 7  
Enter seed: 2  
Unique points generated:  
18 4  
5 17  
0 9  
6 0  
19 9  
16 15  
12 14
```

Enter number of points: 40

Enter seed: 1

Unique points generated:

1 4  
9 19  
8 10  
10 9  
15 10  
2 19  
20 4  
20 7  
3 15  
16 16  
17 14  
12 9  
2 5  
5 13  
1 19  
5 0  
3 12  
17 9  
1 7  
15 18  
12 14  
20 10  
20 2  
2 15  
17 19  
7 8  
8 9  
11 11  
1 10  
9 6  
11 12  
17 5  
19 18  
10 14  
12 3  
9 3  
11 14  
9 14  
7 3  
11 18

Enter number of points: 40

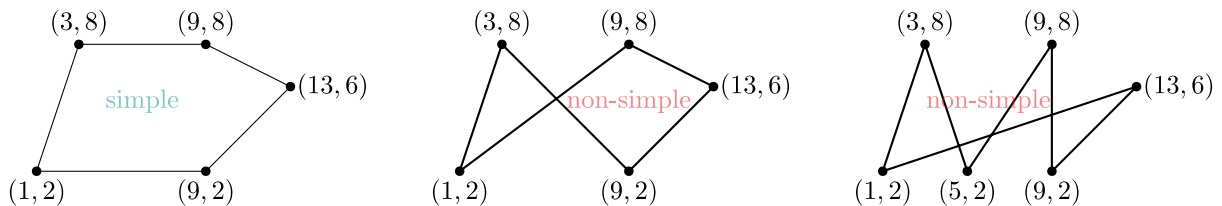
Enter seed: 2

Unique points generated:

18 4  
5 17  
0 9  
6 0  
19 9  
16 15  
12 14  
5 0  
17 3  
12 11  
8 2  
2 11  
4 10  
5 20  
3 9  
9 0  
13 15  
17 12  
1 0  
12 20  
8 5  
19 17  
0 15  
18 10  
5 4  
10 6  
15 14  
14 19  
11 17  
7 0  
15 19  
13 9  
10 14  
8 1  
11 16  
5 2  
15 3  
17 13  
1 20  
16 11

**w06-2.** A polygon is said to be **simple** if its non-adjacent sides do not intersect each other. Write a C program to find out whether an input polygon is simple. It should have the following input/output behavior:

- ◆ The program first takes in the number of vertices  $n$  from the user.
- ◆ Then it takes in  $n$  integer points (i.e., points with integer coordinates) as vertices from the user. The user is assumed to enter these points either in clockwise or in anticlockwise order. In other words, the sides of the polygon connect consecutive vertices in the input sequence (and the last and the first vertices).
- ◆ The program outputs whether or not the polygon is simple. If the polygon is not simple, it also outputs the number of intersection points among the non-adjacent sides.
- ◆ You should write a function that takes as input the endpoints of two straight line segments and returns whether they intersect or not. Call this function from main() to determine the intersection between every two non-adjacent sides of the polygon.



**Sample i/o 1:**

```
How many vertices? 5
Enter X and Y co-ordinates:
1 2
9 2
13 6
9 8
3 8
Polygon is a simple polygon.
```

**Sample i/o 2:**

```
How many vertices? 6
1 2
13 6
9 2
9 8
5 2
3 8
Polygon is a not simple polygon.
Number of intersecting pairs of non-adjacent sides is 3.
```

**Sample i/o 3 (from w06-1 output, seed 1):**

How many vertices? 40

Enter X and Y co-ordinates:

```
1 4
9 19
8 10
10 9
15 10
2 19
20 4
20 7
3 15
16 16
17 14
12 9
2 5
5 13
1 19
5 0
3 12
17 9
1 7
15 18
12 14
20 10
20 2
2 15
17 19
7 8
8 9
11 11
1 10
9 6
11 12
17 5
19 18
10 14
12 3
9 3
11 14
9 14
7 3
11 18
```

Polygon is a not simple polygon.

Number of intersecting pairs of non-adjacent sides is 196.

**Sample i/o 4 (from w06-1 output, seed 2):**

How many vertices? 40

Enter X and Y co-ordinates:

```
18  4
  5 17
  0  9
  6  0
19  9
16 15
12 14
  5  0
17  3
12 11
  8  2
  2 11
  4 10
  5 20
  3  9
  9  0
13 15
17 12
  1  0
12 20
  8  5
19 17
  0 15
18 10
  5  4
10  6
15 14
14 19
11 17
  7  0
15 19
13  9
10 14
  8  1
11 16
  5  2
15  3
17 13
  1 20
16 11
```

Polygon is a not simple polygon.

Number of intersecting pairs of non-adjacent sides is 225.

## Week 9: 1-Feb-2022 (2D Array, Structure.)

### Instructions:

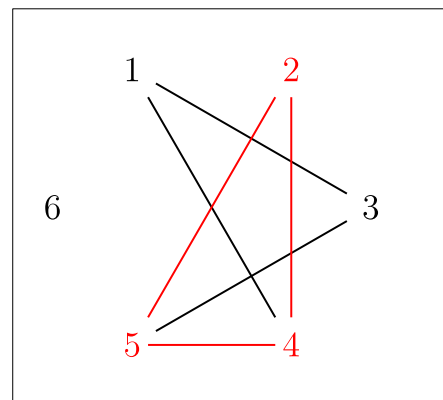
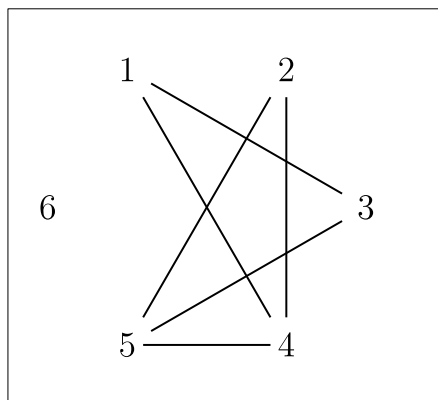
- Use of advanced concepts like pointers and dynamic memory allocation is not permitted and will attract penalty.
- Use of library functions other than printf and scanf is not permitted and will attract penalty.

**w09-1.** There are  $n$  people out of which *any two* people are either friends or strangers. A friendship group is a set of people such that *any two* people in that set are friends. In this problem you will find a largest friendship group.

- ◆ Take  $n$  as input from the keyboard. Assume that  $n$  is a positive integer at most 10. The  $n$  people will be indexed as  $1, \dots, n$ . Also ask the user for the total number of pairs of people who are friends. We will denote that number by  $m$ .
- ◆ The user then enters the  $m$  pairs of people who are friends. This information is to be stored in a 2D binary (integer) array  $A$ . As there are at most 10 people, the size of  $A$  will be  $10 \times 10$ .  $A[i-1][j-1]$  and  $A[j-1][i-1]$  should be 1 if  $i$  and  $j$  are friends and 0 otherwise.
- ◆ Compute and print the people who are part of a largest friendship group. Also print its size.
- ◆ In addition to  $A$ , you may use extra 1D arrays.

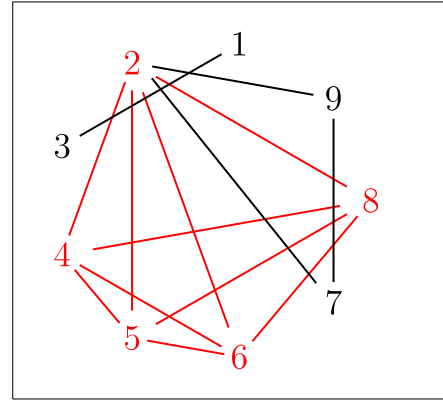
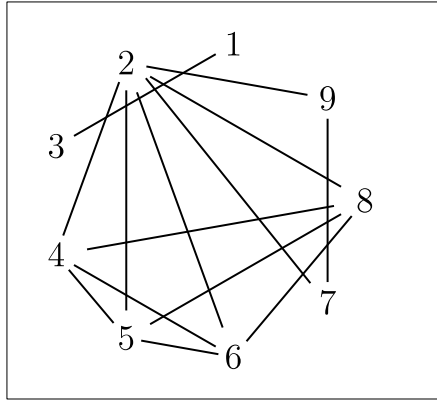
### Sample I/O:

```
How many people and friendships? 6 6
Enter the two guests participating in friendship 1: 1 3
Enter the two guests participating in friendship 2: 1 4
Enter the two guests participating in friendship 3: 3 5
Enter the two guests participating in friendship 4: 2 4
Enter the two guests participating in friendship 5: 4 5
Enter the two guests participating in friendship 6: 5 2
Largest friendship group:
2 4 5
Size=3
```



Largest group is shown in red





Largest group is shown in red

**w09-2.** In this problem you will write a C program that adds two bivariate polynomials  $P_1$  and  $P_2$ . Assume that each polynomial is a sum of terms (monomials) of the form  $c_{ij}x^i y^j$  where  $c_{ij}$  is a non-zero integer,  $i$  and  $j$  are non-negative integers, and  $x$  and  $y$  are variables. Example:  $5x + 2xy^2 - 3y$ . Assume that each polynomial has at most 10 monomials.

- ◆ Define a global structure type with appropriate fields so that a variable of that type can contain a term of the polynomial (i.e., the coefficient and the two exponents).
- ◆ Ask the user for the terms of the polynomial  $P_1$ . Each term is to be entered as a triple  $(i, j, c_{ij})$ . The user can enter the terms in any order, and she finally enters a term with coefficient 0 to indicate that there is no more term to be entered. See sample I/O below.
- ◆ The terms are to be stored in an array of structures in lexicographic order of the corresponding monomials (explained later) as they are entered. At all times, the array should be lexicographically sorted.
- ◆ After storing  $P_1$ , print  $P_1$  in the format shown below: print the terms as bracketed comma-separated triples; every term in a new line.
- ◆ Similarly to  $P_1$ , store the polynomial  $P_2$  from user inputs in another array of structures.
- ◆ Similarly to  $P_1$ , print  $P_2$ .
- ◆ The terms of  $P_1$  and  $P_2$  are to be inserted to the respective arrays by a function call to a function. The actual prototype of the function is left up to you.
- ◆ You should then add the two polynomials and store the result of the addition in another array of structures (that can accommodate at most 20 terms), with the terms of the sum sorted in lexicographic order. You should write a function to add the polynomials.
- ◆ Finally, print the sum polynomial in the format shown in the sample output below.

### **Lexicographic order:**

We will say that a monomial  $x^i y^j$  is lexicographically smaller than another monomial  $x^k y^\ell$  if either  $i < k$  or  $(i = k \text{ and } j < \ell)$ . Example:  $x^2 y < x^3$  and  $x^2 y^3 < x^2 y^4$ .

**Example 1:**

$$P_1 = -6xy^2 + 4xy^3 + 8x^2y^5, \quad P_2 = 6xy^2 + x^2y^5 + 8x^3y^5 - x^4y^4.$$

$$\text{Sum} = 4xy^3 + 9x^2y^5 + 8x^3y^5 - x^4y^4.$$

**Sample I/O**

Input polynomial 1:

Enter exponent of x, exponent of y and coefficient: 1 3 4

Enter exponent of x, exponent of y and coefficient: 2 5 8

Enter exponent of x, exponent of y and coefficient: 1 2 -6

Enter exponent of x, exponent of y and coefficient: 4 2 0

Polynomial 1:

(1, 2, -6)

(1, 3, 4)

(2, 5, 8)

Input polynomial 2:

Enter exponent of x, exponent of y and coefficient: 3 5 8

Enter exponent of x, exponent of y and coefficient: 1 2 6

Enter exponent of x, exponent of y and coefficient: 2 5 1

Enter exponent of x, exponent of y and coefficient: 4 4 -1

Enter exponent of x, exponent of y and coefficient: 9 9 0

Polynomial 2:

(1, 2, 6)

(2, 5, 1)

(3, 5, 8)

(4, 4, -1)

Sum:

(1, 3, 4)

(2, 5, 9)

(3, 5, 8)

(4, 4, -1)

**Example 2:**

$$P_1 = 5 - 4y - 3x - 5xy + 4xy^2 + 6x^2 - 8x^2y^2 - 2x^4, \quad P_2 = 2y - x + xy - 2x^2 + x^4y^4.$$

$$\text{Sum} = 5 - 2y - 4x - 4xy + 4xy^2 + 4x^2 - 8x^2y^2 - 2x^4 + x^4y^4.$$

## Week 10: 15-Feb-2022 (Strings)

**w10-1.** In this problem, you will scan a poem (along with its name and its poet) and do some computations as follows.

- ◆ Take a poem as input through the keyboard and store it in a string variable. Assume that the poem has at most 1000 characters in total. Use `scanf ("%^[^#]s", P)` for scanning the whole text at once in the character array `P`.  
The poem has the following form.
  - Line 1 contains the name of the poem.
  - Line 2 is blank (= a single newline character).
  - Line 3 contains the poet's name.
  - Line 4 is blank.
  - The poem starts from Line 5.
  - There are no two consecutive blank lines anywhere.
  - Stanzas are separated by exactly one blank line.
  - After the last line of the poem, the character '#' appears in a new line to mark the end.
  - There is exactly one space or a punctuation mark or a newline after each word.
  - There is no unnecessary space anywhere.
  - A punctuation mark is immediately preceded by a word, and immediately followed by either a space or a newline.
  - A hyphenated word is counted as a single word (ex: fast-paced, I-). A hyphen is not counted as a punctuation mark.
  - Following are to be counted as punctuation marks: '.', ',', ':', ';', '!', '?'.
  - A word with apostrophe is counted as a single word (ex: we've, there's).
- ◆ Print the name of the poem and its poet **[2 marks]**, and the number of lines **[2 marks]**, words **[2 marks]**, stanzas **[2 marks]**, and punctuation marks **[2 marks]** in the main body of poem.

### Example 1:

Enter a poem with its name and poet:

What a Life!

Lincoln, 11, from the United Kingdom

Life was always fast-paced, we never slowed down,  
Until everything stopped when Corona came to town.

Now all is quiet and there's peace all around,  
We've looked in our hearts and kindness we've found.  
#

Poem name: What a Life!

Poet name: Lincoln, 11, from the United Kingdom

Lines: 4

Words: 34

Stanzas: 2

Punctuation marks: 5

### Example 2:

Enter a poem with its name and poet:  
The Road Not Taken

Robert Frost

Two roads diverged in a yellow wood,  
And sorry I could not travel both  
And be one traveler, long I stood  
And looked down one as far as I could  
To where it bent in the undergrowth;

Then took the other, as just as fair,  
And having perhaps the better claim,  
Because it was grassy and wanted wear;  
Though as for that the passing there  
Had worn them really about the same,

And both that morning equally lay  
In leaves no step had trodden black.  
Oh, I kept the first for another day!  
Yet knowing how way leads on to way,  
I doubted if I should ever come back.

I shall be telling this with a sigh  
Somewhere ages and ages hence:  
Two roads diverged in a wood, and I-  
I took the one less traveled by,  
And that has made all the difference.  
#

Poem name: The Road Not Taken  
Poet name: Robert Frost

Lines: 20  
Words: 144  
Stanzas: 4  
Punctuation marks: 17

### Example 3:

Enter a poem with its name and poet:  
What a Life!

Lincoln, 11, from the United Kingdom

Life was always fast-paced, we never slowed down,  
Until everything stopped when Corona came to town.

Now all is quiet and there's peace all around,  
We've looked in our hearts and kindness we've found.

We learn now with mum, this is a new feature,

But we can't wait to get back to our teacher.

I miss Sea Cadets, school, my friends and my dad,  
I miss sharing the fun times and that makes me sad.

We've had social distancing picnics, social distancing walks,  
Social distancing hugs and social distancing talks.

I'm looking forward to getting away,  
The beach, the hotel and a perfect holiday.

When it is? I'll throw my arms open wide,  
And shout to the world, WE CAN ALL GO OUTSIDE!

Don't give up hope, the end is in sight,  
If we all stick together, we'll all win this fight.  
#

Poem name: What a Life!

Poet name: Lincoln, 11, from the United Kingdom

Lines: 16

Words: 142

Stanzas: 8

Punctuation marks: 26

- w10-2.** In this problem you will rearrange an input shuffle of playing cards in a definite order. You might know that playing cards have 4 suits, each having 13 cards with unique numbers. The ordered suits are ♣ Clubs (C), ♦ Diamonds (D), ♥ Hearts (H), and ♠ Spades (S), and the ordered numbers are 2, 3, ..., 9, J, Q, K, A. (10 is not considered for this problem.) So, in total we have 48 cards, each being uniquely identified by its suit and number.  
For your code, proceed as follows.

- ◆ Define a global structure `card` that can store the ID (= suit and number in char) of a card.
- ◆ Define in `main()` an array of structure `card` having size 48.
- ◆ Take in a sequence of cards as input, and store the sequence in a string variable; for example, `scanf("%[^\\n]s", s)` to store it in `s`. Each card should be a pair of characters (e.g., 4C stands for 4 of Clubs, AD stands of Ace of Diamonds, etc.). Thus, the first character in the pair is the number and the second is the suit. The sequence should be a space-separated sequence of such pairs. You may assume that the cards are all distinct.
- ◆ Extract the card information from the string and populate the array of `cards`.
- ◆ Card ordering: Principle is *suits first, numbers next*. Find the smallest card and exchange it with the 1st card. Now, from the last  $n - 1$  cards, find their smallest and exchange it with their 1st card. Next, from the last  $n - 2$  cards, find their smallest and exchange it with their 1st card. And so on.
- ◆ Print the ordered cards. While printing, you should print the actual symbols of the suits. That is done as follows. Declare globally the following strings:

```
#if defined(_WIN32) || defined(__MSDOS__)
#define spade    "\x06"
#define club    "\x05"
#define heart    "\x03"
#define diamond "\x04"
#else
```

```

#define spade    "\xe2\x99\xa0"
#define club     "\xe2\x99\xa3"
#define heart    "\xe2\x99\xa5"
#define diamond  "\xe2\x99\xa6"
#endif

```

If you print the first string by using `printf("%s", club)`, then the symbol of clubs will be printed. Similarly for other suits. A file named `print-card-suits-in-black-all-compilers.c` is available in moodle, you can first download, compile and run it.

### How it works:

```

7D 3S AC 9C QS - smallest among last 5 cards = 9C
9C 3S AC 7D QS - smallest among last 4 cards = AC
9C AC 3S 7D QS - smallest among last 3 cards = 7D
9C AC 7D 3S QS - smallest among last 2 cards = 3S
9C AC 7D 3S QS - stop as there is just one card left.

```

### Sample I/O:

Enter the sequence of cards:

7D 3S 9C

After ordering:

9♣ 7♦ 3♠

Enter the sequence of cards:

7D 3S AC 9C QS

After ordering:

9♣ A♣ 7♦ 3♠ Q♠

Enter the sequence of cards:

7D 3H AC 9C JH KD 3D 2C AS QC 5S 4D 3S

After ordering:

2♣ 9♣ Q♣ A♣ 3♦ 4♦ 7♦ K♦ 3♥ J♥ 3♠ 5♠ A♠

Enter the sequence of cards:

JC 4D 3D 5H 9S 2C 2D 4S KD QC QD 9H 8H

After ordering:

2♣ J♣ Q♣ 2♦ 3♦ 4♦ Q♦ K♦ 5♥ 8♥ 9♥ 4♠ 9♠

## Week 11: 22-Feb-2022 (Pointers, parameter passing, string library functions, searching, sorting.)

**w11-1.** In this program you will sort a list of student records.

- ◆ Declare a global structure named `student` in `main()` that can store the roll number and the name of a student. Assume that the name consists of at most 99 English letters, and the roll number is a combination of 9 digits and English letters as we see in IITKGP.
- ◆ Ask the user for the number of students. Then dynamically allocate an array of structure `student` of appropriate size. Populate it with user input. Use `scanf("%[^\n]s"...)` for scanning the name.
- ◆ Ask the user to enter 1 to sort the array by roll number, and 0 to sort the array by name.
- ◆ Use bubble sort to sort the array. The order should be the lexicographic order of the respective string. To swap two structure variables in the list, you must use a function with the following prototype:

```
void swap(struct student *A, struct student *B);
```

- ◆ Print the sorted student records as shown in the examples below.

### Example 1 (file `stu-1.txt`)

```
How many students? 15
```

```
Enter roll numbers and names:
```

```
21BT10017 Jay Godara
17CS30042 Amatya Sharma
16CS91R03 Souvik Sur
21BT30001 Abhinay Kumar Pandey
21AG30022 Kushal
17CS30030 Sanket Meshram
17CS30033 Shivam Kumar Jha
21AG30042 Suryaansh Jindal
21AE10038 Subhadip Murmu
21AE30015 Paridhi D Choudhary
21CE10067 Sreejan Shivam
21CE30002 Abhijeet Nigam
21CE30022 Prabhat Kumar
17CS30034 Shrey Shrivastava
19CS91R09 Shankho Subhra Pal
```

```
Enter 1 to sort by roll number, 0 to sort by name: 1
```

```
After sorting by roll number:
```

```
1. 16CS91R03 Souvik Sur
2. 17CS30030 Sanket Meshram
3. 17CS30033 Shivam Kumar Jha
4. 17CS30034 Shrey Shrivastava
5. 17CS30042 Amatya Sharma
6. 19CS91R09 Shankho Subhra Pal
7. 21AE10038 Subhadip Murmu
8. 21AE30015 Paridhi D Choudhary
9. 21AG30022 Kushal
```

```
10. 21AG30042  Suryaansh Jindal
11. 21BT10017  Jay Godara
12. 21BT30001  Abhinay Kumar Pandey
13. 21CE10067  Sreejan Shivam
14. 21CE30002  Abhijeet Nigam
15. 21CE30022  Prabhat Kumar
```

### Example 2 (file stu-1.txt)

```
How many students? 15
Enter roll numbers and names:
```

```
21BT10017 Jay Godara
17CS30042 Amatya Sharma
16CS91R03 Souvik Sur
21BT30001 Abhinay Kumar Pandey
21AG30022 Kushal
17CS30030 Sanket Meshram
17CS30033 Shivam Kumar Jha
21AG30042 Suryaansh Jindal
21AE10038 Subhadip Murmu
21AE30015 Paridhi D Choudhary
21CE10067 Sreejan Shivam
21CE30002 Abhijeet Nigam
21CE30022 Prabhat Kumar
17CS30034 Shrey Shrivastava
19CS91R09 Shankho Subhra Pal
```

```
Enter 1 to sort by roll number, 0 to sort by name: 0
After sorting by name:
```

```
1. 21CE30002  Abhijeet Nigam
2. 21BT30001  Abhinay Kumar Pandey
3. 17CS30042  Amatya Sharma
4. 21BT10017  Jay Godara
5. 21AG30022  Kushal
6. 21AE30015  Paridhi D Choudhary
7. 21CE30022  Prabhat Kumar
8. 17CS30030  Sanket Meshram
9. 19CS91R09  Shankho Subhra Pal
10. 17CS30033  Shivam Kumar Jha
11. 17CS30034  Shrey Shrivastava
12. 16CS91R03  Souvik Sur
13. 21CE10067  Sreejan Shivam
14. 21AE10038  Subhadip Murmu
15. 21AG30042  Suryaansh Jindal
```

**w11-2.** In this program, you will search for a *prefix* in a list of student records sorted by roll numbers. (Definition of prefix will be explained. Ex: The string “abc” has three prefixes – “a”, “ab”, “abc”.)

- ◆ Define a global structure named `student` as given in the first problem.
- ◆ Ask the user for the number of student records. Then dynamically allocate an array of structures of appropriate size.



- ◆ Populate it with user input. Assume that the user enters student records in increasing (lexicographic) order of the roll numbers.
- ◆ The user should then enter a search string *s*.
- ◆ The program should then use binary search to search for and print all student records that have *s* as a prefix.

### Example 1 (file `stu-2-sorted-by-roll.txt`)

```
How many students? 200
Enter roll numbers and names (sorted by roll number):
16CS91R03  Souvik Sur
17CS30001  Abhik Naskar
17CS30002  Adhikansh Singh
17CS30003  Alpesh Kaushal
17CS30004  Ankit Saurabh
17CS30005  Anshul Goel
17CS30006  Anunay Sharma
17CS30007  Apoorve Singhal
17CS30008  Arnab Maiti
17CS30009  Bagde Ankit
17CS30010  Bajarua Sashank Srivardhan
17CS30011  Deepank Agrawal
17CS30012  Dewang Modi
17CS30014  Vasu
17CS30015  Hariyala Omkara Naga Sai Varshith
17CS30016  Harsh Pritam Sanapala
17CS30017  Deepak Imandi
17CS30018  Kanishk Singh
17CS30020  Snehal Reddy
17CS30021  Kshitiz Sharma
17CS30024  Nari Rohini
17CS30025  Pericherla Amshumaan Varma
17CS30027  Ritikesh Gupta
17CS30028  Rohit
17CS30029  Rohit Pathak
17CS30030  Sanket Meshram
17CS30031  Sashank Bonda
17CS30033  Shivam Kumar Jha
17CS30034  Shrey Shrivastava
17CS30035  Siddhant Agarwal
17CS30037  Venktesh Lagaskar
17CS30040  Yogesh Kumar
17CS30041  Aadarsh Sahoo
17CS30042  Amatya Sharma
17CS30042  Amatya Sharma
17CS30043  Faraaz Rahman Mallick
17CS30044  Udit Desai
17CS91P02  Bijoy Das
17CS91P07  Sayandeep Sanyal
17CS91P08  Soumi Das
17CS91R03  Arnab Bag
17CS91R06  Indrajit Mazumdar
17CS91R10  Soumya Majumdar
```

17CS92R03 Haque Arijul  
18CS72P07 Sk Mainul Islam  
18CS91P01 Anirban Chakraborty  
18CS91P04 Debranjana Pal  
18CS91P05 Durba Chatterjee  
18CS92P02 Anurag Roy  
18CS92R02 Biswajeet Sethi  
18CS92R03 Dev Narayan Yadav  
18CS92R08 Sourav Biswas  
18CS92R08 Sourav Biswas  
19CS71P02 Siddhartha Chowdhury  
19CS72P03 Bharathi Chaudhury  
19CS72P07 Soumyajit Das  
19CS90J01 Sumanta Dey  
19CS90J02 Briti Gangopadhyay  
19CS91F01 Bishnu Charan Behera  
19CS91F02 Soumen Paul  
19CS91P01 Akashdeep Saha  
19CS91P02 Soumyadyuti Ghosh  
19CS91R04 Manoranjan Behera  
19CS91R06 Preetam Chayan Chatterjee  
19CS91R07 Rijoy Mukherjee  
19CS91R09 Shankho Subhra Pal  
19CS91R12 Vasudha Joshi  
19CS92F01 Riya Samanta  
19CS92P01 Kounteya Sarkar  
19CS92P02 Kuheli Pratihara  
19CS92R05 Bithika Pal  
19CS92R06 Punyajoy Saha  
19ET91R01 Jyoti Prakash Meher  
19cs92r04 Sourav Das  
20CS60D01 Amit Barik  
20CS60R03 Abhishek Mukherjee  
20CS60R04 Satyaki Das  
20CS60R06 Sneha Jogdhankar  
20CS60R07 Swarnava Das  
20CS60R08 Susmita Paul  
20CS60R09 Haresh Ramdas Gaikwad  
20CS60R12 Pratibha Singh  
20CS60R13 Rutwik Pandit  
20CS60R14 Kirti Agarwal  
20CS60R15 Sontu Mistry  
20CS60R16 Akhil Jaiswal  
20CS60R18 Suyash Tiwari  
20CS60R19 Subham Jana  
20CS60R20 Md Laadla  
20CS60R23 Sreyasree Mandal  
20CS60R24 Aditya Anand  
20CS60R24 Aditya Anand  
20CS60R26 Aditya Devasthale  
20CS60R27 Sanjay Jaiswal  
20CS60R28 Manasvi Sagarkar  
20CS60R29 Ved Prakash  
20CS60R30 Sagar Gupta

20CS60R31 Akash Kumar Gangwar  
20CS60R34 Shreyas More  
20CS60R36 Soumya Porel  
20CS60R37 Vipray Jain  
20CS60R38 Rishabh Sanjeev Thakur  
20CS60R40 Akash Singh Sant  
20CS60R42 Mehul Vaidya  
20CS60R43 Shubham Patidar  
20CS60R44 Adesh Sharma  
20CS60R45 Venu Gopal Bandhakavi  
20CS60R46 Deepanshi Pandey  
20CS60R47 Manjeet Kumar  
20CS60R48 Ashish Kumar Singh  
20CS60R50 Sandeep Kumar Shahu  
20CS60R52 Yogesh Porwal  
20CS60R56 Vineeth Kumar Balapanuru  
20CS60R58 Km Simran Jaiswal  
20CS60R60 Ravi Pratap Singh  
20CS60R64 Sahil Jain  
20CS60R65 Saurav Koranga  
20CS60R65 Saurav Koranga  
20CS60R65 Saurav Koranga  
20CS60R70 Ram Kishor Yadav  
20CS60R71 Rohit  
20CS71P03 Sukanya Das  
20CS91F01 Das Adhikary Debjyoti Tushar  
20CS91J04 Sugandh Pargal  
20CS91J05 Somnath Hazra  
20CS91P02 Deepak Mewada  
20CS91Q01 Danny Pereira  
20CS91R02 Abhishek Kumar  
20CS91R07 Arup Kumar Dutta  
20CS91R08 Chandan Kumar  
20CS91R09 Kiran Purohit  
20CS91R09 Kiran Purohit  
20CS91R10 Koustav De  
20CS91R12 Purnima Gautam  
20CS91R13 Tapadyoti Banerjee  
20CS91R14 Soham Poddar  
20CS92P01 Atonu Ghosh  
20CS92P05 Argha Sen  
20CS92R01 Ankita Saha  
20CS92R05 Siddharth D Jaiswal  
21AE10038 Subhadip Murmu  
21AE30015 Paridhi D Choudhary  
21AG30022 Kushal  
21AG30042 Suryaansh Jindal  
21BT10017 Jay Godara  
21BT30001 Abhinay Kumar Pandey  
21CE10067 Sreejan Shivam  
21CE30002 Abhijeet Nigam  
21CE30022 Prabhat Kumar  
21CS91R14 Sunandan Adhikary  
21EC10006 Anushka Kishor Susatkar

21EC10026 Gaykwad Shreyansh Natha  
21EC10046 Mohan S  
21EC10066 Shaily Jain  
21EC10086 Veeransh Manish Mehta  
21EC30018 Dhruv Sarkar  
21EC30038 Polavarapu Soumya  
21EC30058 Venkata Krishna Aditya Arigila  
21EE10019 Arkadipta Saha  
21EE10039 Kashish Morey  
21EE10059 S Arulanandan  
21EE10079 Yashwanth Gowda G S  
21EE30020 Samarth Adatia  
21EX10007 Atharva Chilwarwar  
21EX10027 Rajarshi Biswas  
21GG10005 Arjun Singh  
21GG10025 Md Dilshad Alam Asrafi  
21GG10045 Voonna Sanjana  
21HS10013 Anshul Raj  
21HS10032 Paridhi Chaurasia  
21HS10051 Utkarsh Verma  
21IE10012 Debargha Das  
21IE10031 Rathod Rajkumar  
21IM10006 Anurag Saha  
21IM10025 Rishabh Prasad  
21IM30002 Aditya Choudhary  
21IM30021 Sanjeev Mallick  
21MA10012 Anushka Aashvi  
21MA10031 Manas Jha  
21MA10050 Raunaq Bose  
21MA20003 Vanshika Bal  
21ME10019 Bhanu Pratap Singh Rathore  
21ME10038 Hrishikesh M V  
21ME10057 Pratham Vishwakarma  
21ME10076 Shaunak Ghosh  
21ME10095 Vikalp Borkar  
21ME30010 Alakesh Baro  
21ME30029 Harshvardhan Acharya  
21ME30048 Ojasvi R Singh  
21ME30067 Tejal Balyan  
21MF10012 Binamrata Mandal  
21MF10031 Sagar Bapu Raykar  
21MF3IM07 Jambarapu Wesly  
21MI10005 Akshay P Sawai  
21MI10024 Harshit Kumar Singh  
21MI10043 Rohan Kumar  
21MI31008 Ayush Kumar  
21PH10009 Bodasakurti Venkat Sai Dutt  
21PH10028 Prabaha Das  
21PH20002 Maloth Maniteja

Enter prefix: 17CS91

Found:

17CS91P02 Bijoy Das

```
17CS91P07 Sayandeep Sanyal
17CS91P08 Soumi Das
17CS91R03 Arnab Bag
17CS91R06 Indrajit Mazumdar
17CS91R10 Soumya Majumdar
```

### **Example 2 (file stu-2-sorted-by-roll.txt)**

```
How many students? 200
Enter roll numbers and names (sorted by roll number):
16CS91R03 Souvik Sur
17CS30001 Abhik Naskar
...
21PH10028 Prabaha Das
21PH20002 Maloth Maniteja

Enter prefix: 21MA
```

```
Found:
21MA10012 Anushka Aashvi
21MA10031 Manas Jha
21MA10050 Raunaq Bose
21MA20003 Vanshika Bal
```

### **Example 3 (file stu-2-sorted-by-roll.txt)**

```
How many students? 200
Enter roll numbers and names (sorted by roll number):
16CS91R03 Souvik Sur
17CS30001 Abhik Naskar
...
21PH10028 Prabaha Das
21PH20002 Maloth Maniteja

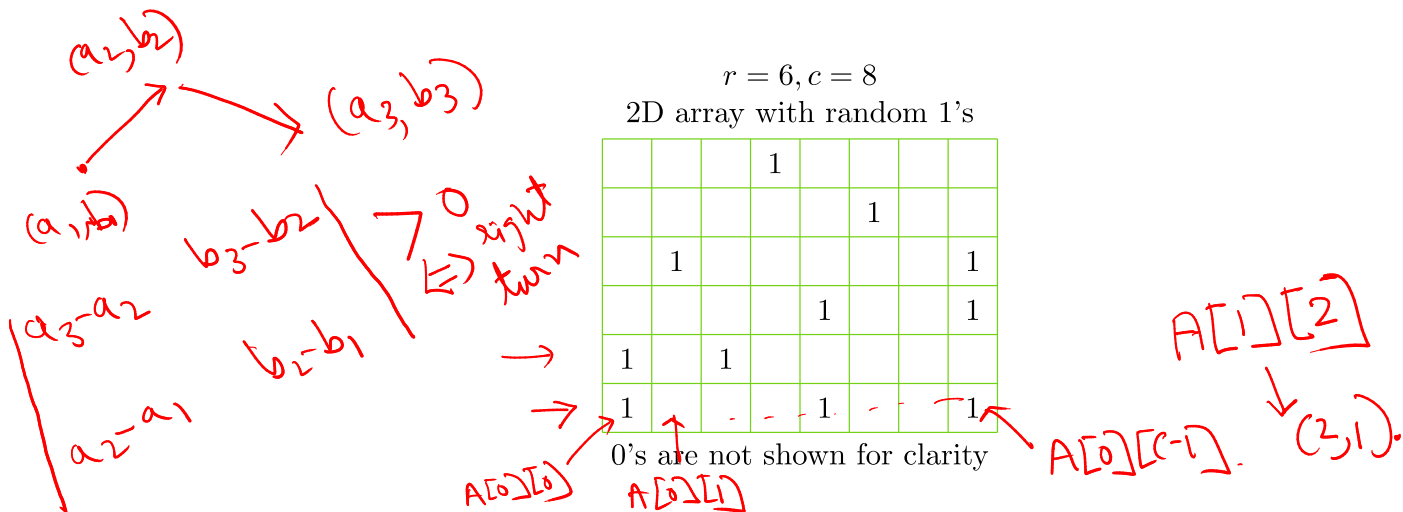
Enter prefix: 21MB
None found!
```

## Week 13: 08-March-2022 (Dynamic memory allocation of 2D array, linked list, stack, queue.)

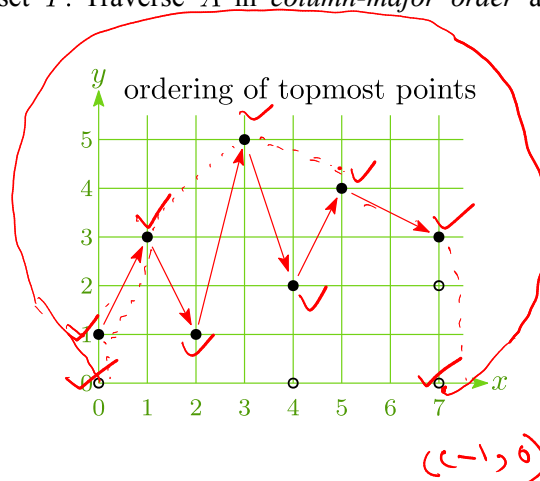
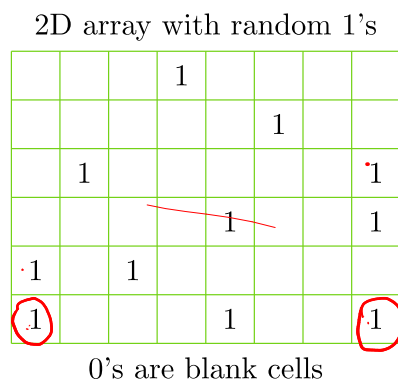
w13-1. This is the only question of today.

- a) With three positive numbers  $r, c, t$  as input from the user, dynamically allocate a 2D character array  $A$  with  $r$  rows and  $c$  columns. Assign  $A[0][c-1] = A[0][0] = 1$ , and fill up the rest of  $A$  randomly with 0's and 1's. To control the number of 1's in  $A$ , you need  $t$ . Assume that the value of  $t$  lies in the interval  $[1, 100]$ . Use random number generator (see w06-1) to generate random numbers in  $[1, 100]$  for populating the array based on the value of  $t$ . The probability that a particular entry is 1 should be  $t/100$ .

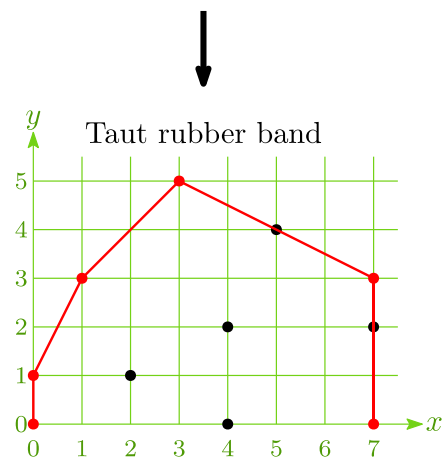
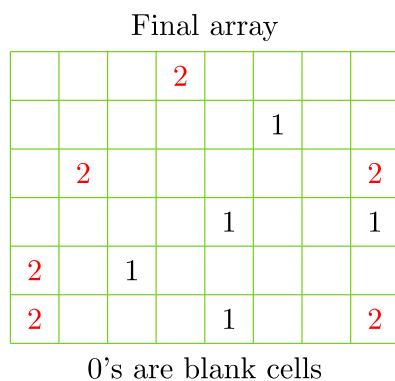
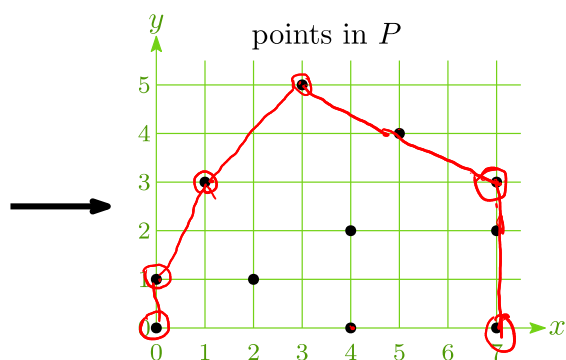
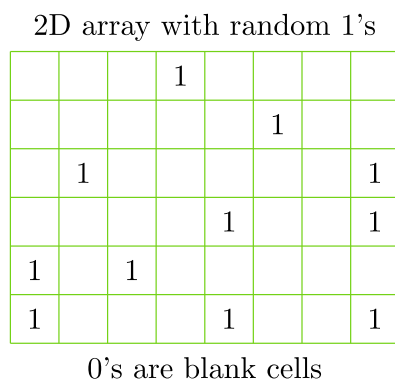
Print the binary array  $A$  on the terminal. For clarity, print 0's as hyphens. [5 marks]



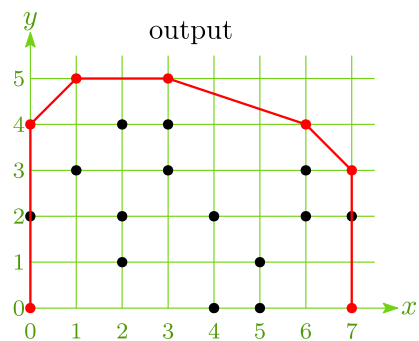
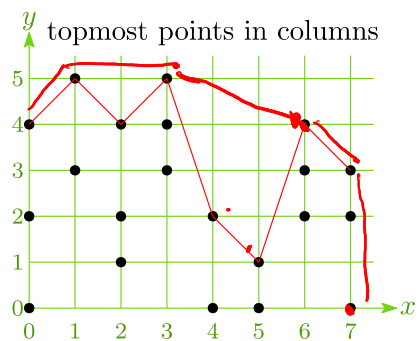
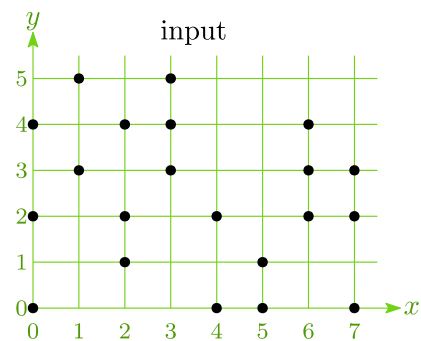
- b) Use the rows and columns of  $A$  as the  $(y, x)$ -coordinates of the points corresponding to the 1's in  $A$ . Mind that  $x$  corresponds to column, whereas  $y$  to row. Consider the points  $(0, 0)$ ,  $(c-1, 0)$ , and the topmost points of the columns; call this set  $P$ . Traverse  $A$  in *column-major order* and print  $P$  in increasing order of  $x$ . [5 marks]

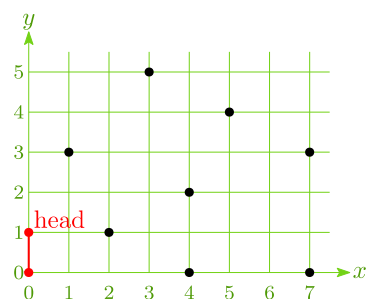


- c) Find the **taut rubber band**  $R$  (as a piecewise-linear poly-line) from  $(0, 0)$  to  $(c-1, r-1)$ , and passing from the first to the last point of  $P$ , so that no point of  $P$  lies above  $R$ . Mind that the rubber band *always remains taut* and so there is always a right turn at every vertex of  $R$ , excepting its two endpoints. Print the vertices of  $R$  on the terminal and mark their corresponding 1's in  $A$  as '2'. Print the array  $A$  (with 0's as hyphens, for clarity). [5+5 marks]



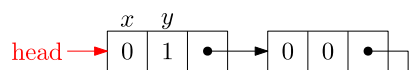
### Another example:



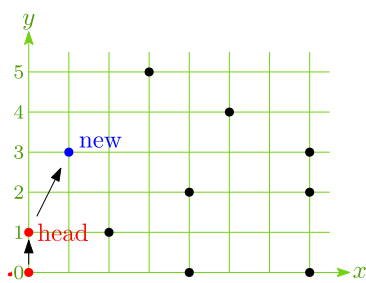


red points are in linked list

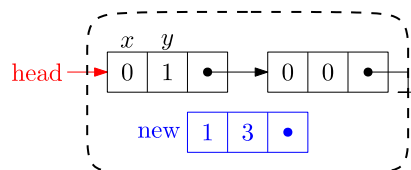
## Linked list



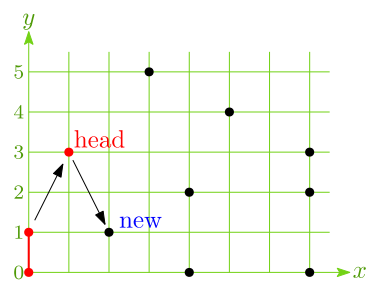
Initialization



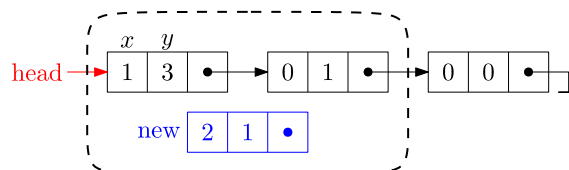
red points are in linked list



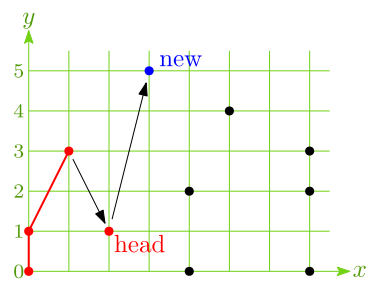
Turn = right  $\implies$  head point is a vertex (so far).  
So just insert the new node.



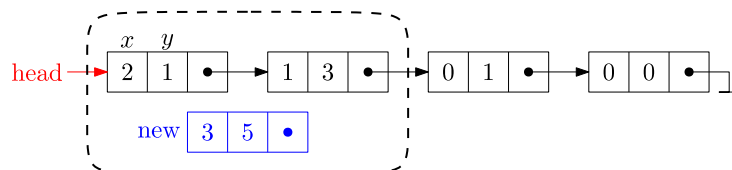
red points are in linked list



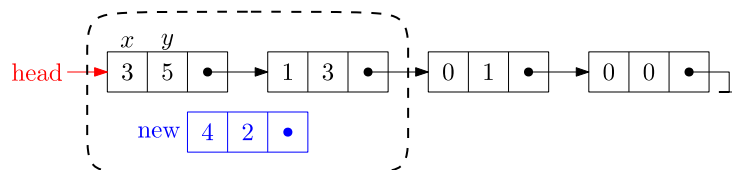
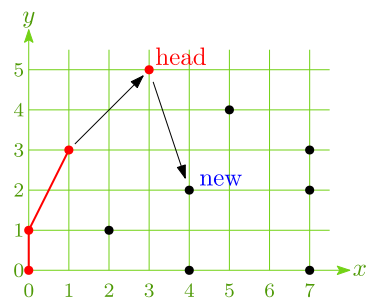
Turn = right  $\implies$  head point is a vertex (so far).  
So just insert the new node.



red points are in linked list

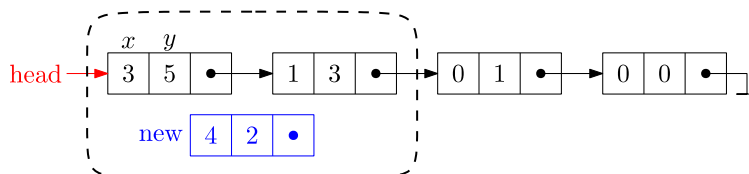
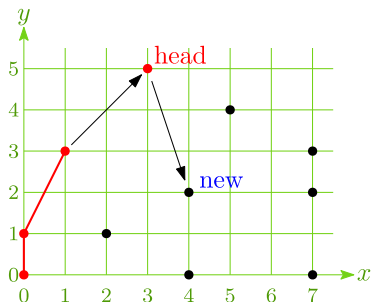


Turn  $\neq$  right  $\implies$  head point  $\neq$  vertex.  
So delete head and insert the new node.

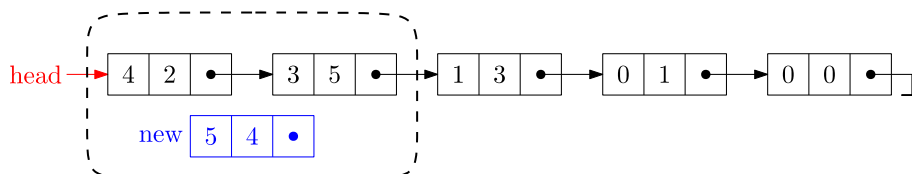
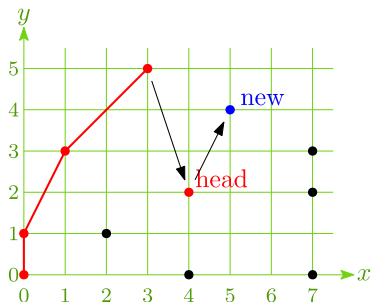


Turn = right  $\implies$  head point is a vertex (so far).  
So just insert the new node.

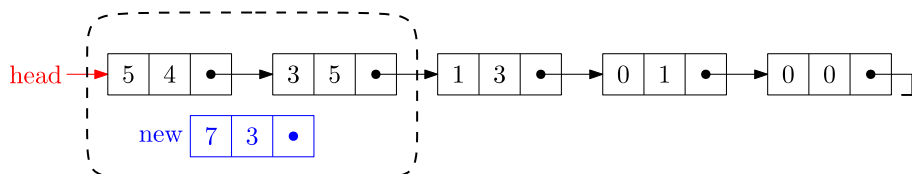
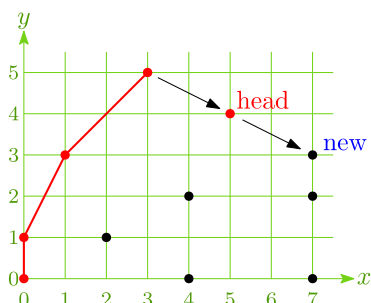




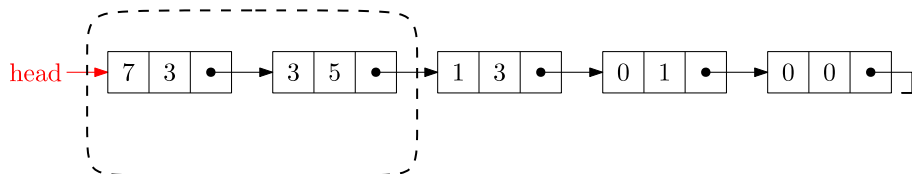
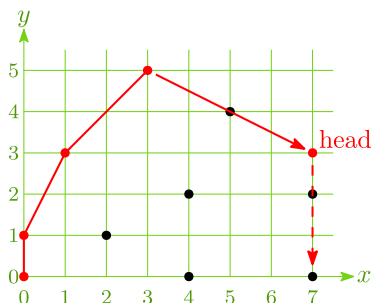
Turn = right  $\implies$  head point is a vertex (so far).  
So just insert the new node.



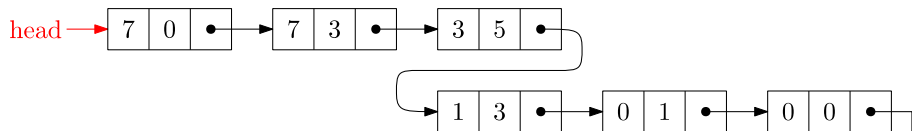
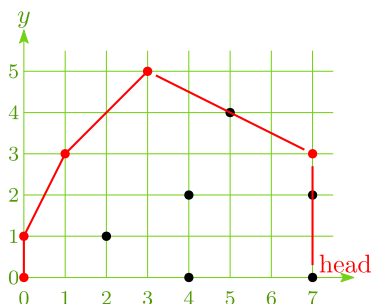
Turn  $\neq$  right  $\implies$  head point  $\neq$  vertex.  
So delete head and insert the new node.



Turn  $\neq$  right  $\implies$  head point  $\neq$  vertex.  
So delete head and insert the new node.



End of turn checking.  
Now insert the endpoint (7,0), as head point is different.



Final linked list.

## Sample input/output:

### Sample 1

Enter no. of rows, no. of columns and expected percentage of 1's: 5 7 20

Enter seed: 1

Input array:

```
- - - - - 1 -  
- 1 - - - - -  
- - - - - - -  
- - - - - - -  
1 - - - 1 - 1
```

List of points:

(0, 0) (1, 3) (4, 0) (5, 4) (6, 0)

Vertices:

(6, 0) (5, 4) (1, 3) (0, 0)

Output array:

```
- - - - - 2 -  
- 2 - - - - -  
- - - - - - -  
- - - - - - -  
2 - - - 1 - 2
```

---

**Sample 2** (same data as Sample 1, only seed is different)

Enter no. of rows, no. of columns and expected percentage of 1's: 5 7 20

Enter seed: 2

Input array:

```
1 - - - 1 - -  
- 1 - - 1 - -  
- - - - - - -  
- - - - - - -  
1 - 1 - - - 1
```

List of points:

(0, 0) (0, 4) (1, 3) (2, 0) (4, 4) (6, 0)

Vertices:

(6, 0) (4, 4) (0, 4) (0, 0)

Output array:

```
2 - - - 2 - -  
- 1 - - 1 - -  
- - - - - - -  
- - - - - - -  
2 - 1 - - - 2
```

---

**Sample 3** (same data as Sample 1, only percentage is different)

Enter no. of rows, no. of columns and expected percentage of 1's: 5 7 40

Enter seed: 1

Input array:

```
- 1 - 1 1 1 1
1 1 - - 1 - 1
- - - 1 1 1 -
1 - - - 1 - 1
1 - - - 1 - 1
```

List of points:

(0, 0) (0, 3) (1, 4) (3, 4) (4, 4) (5, 4) (6, 4) (0, 6)

Vertices:

(6, 0) (6, 4) (1, 4) (0, 3) (0, 0)

Output array:

```
- 2 - 1 1 1 2
2 1 - - 1 - 1
- - - 1 1 1 -
1 - - - 1 - 1
2 - - - 1 - 2
```

---

#### Sample 4

Enter no. of rows, no. of columns and expected percentage of 1's: 10 20 15

Enter seed: 2

Input array:

```
- - - - - 1 1 1 - 1 - - - - -
- - - - - - - - - - - - - - -
1 - - - - - - 1 - 1 - 1 - 1 - - -
- - - - 1 - 1 1 - - - - - - - -
1 - 1 - - - 1 - - - 1 - - - 1 - - -
- 1 - - - - - 1 - - - - - - - -
- 1 - - - - - - - - - - 1 - - - -
- - - - - 1 - - - 1 - - - - - 1 -
- - 1 - - 1 - - 1 - - - 1 - - 1 - -
1 - - - - - - - - - - - - - - 1
```

List of points:

(0, 0) (0, 7) (1, 4) (2, 5) (4, 6) (5, 2) (6, 6) (7, 6) (8, 9) (9, 9) (10, 9) (12, 9) (14, 7) (15, 1) (16, 7) (18, 2) (19, 0)

Vertices:

(19, 0) (16, 7) (12, 9) (8, 9) (0, 7) (0, 0)

Output array:

```
- - - - - 2 1 1 - 2 - - - - -
- - - - - - - - - - - - - - -
2 - - - - - - - 1 - 1 - 1 - 2 - - -
- - - - 1 - 1 1 - - - - - - - -
1 - 1 - - - 1 - - - 1 - - - 1 - - -
- 1 - - - - - 1 - - - - - - - -
- 1 - - - - - - - - - - 1 - - - -
- - - - - 1 - - - 1 - - - - - 1 -
- - 1 - - 1 - - 1 - - - 1 - - 1 - -
2 - - - - - - - - - - - - - - 2
```

---

**Sample 5** (same data as Sample 4, only percentage is different)

Enter no. of rows, no. of columns and expected percentage of 1's: 10 20 30

Enter seed: 2

Input array:

```
- - - - - 1 1 1 - 1 - 1 - - 1 1 1
- - - 1 - - - - 1 - 1 - - - - 1 - - 1
1 - - - - - 1 - - 1 1 1 1 - 1 1 1 - - -
1 - - - 1 - 1 1 - - - - - - - - 1 -
1 - 1 1 - 1 1 - - - 1 - - - 1 - - - -
- 1 - - - - - - 1 - - - 1 1 - - - - -
- 1 - - - 1 1 - 1 - - - 1 - 1 - - - -
- - - - 1 1 - - - 1 - - - - - - - 1 1
- 1 1 - - 1 - - 1 1 - - 1 - - 1 - - -
1 - 1 - - - - - - 1 - - - - - - - 1
```

List of points:

(0, 0) (0, 7) (1, 4) (2, 5) (3, 8) (4, 6) (5, 5) (6, 7) (7, 6) (8, 9) (9, 9) (10, 9) (11, 7) (12, 9) (13, 4) (14, 9) (15, 7) (16, 8) (17, 9) (18, 9) (19, 9) (0, 19)

Vertices:

(19, 0) (19, 9) (8, 9) (3, 8) (0, 7) (0, 0)

Output array:

```
- - - - - 2 1 1 - 1 - 1 - - 1 1 2
- - - 2 - - - - 1 - 1 - - - - 1 - - 1
2 - - - - - 1 - - 1 1 1 1 - 1 1 1 - - -
1 - - - 1 - 1 1 - - - - - - - - 1 -
1 - 1 1 - 1 1 - - - 1 - - - 1 - - - -
- 1 - - - - - - 1 - - - 1 1 - - - - -
- 1 - - - 1 1 - 1 - - - 1 - 1 - - - -
- - - - 1 1 - - - 1 - - - - - - - 1 1
- 1 1 - - 1 - - 1 1 - - 1 - - 1 - - -
2 - 1 - - - - - - 1 - - - - - - - 2
```