PREDICT THE OUTPUT

A.

```
1.   #include <stdio.h>
2.   void foo(int*);
3.   int main(){
4.       int i = 10;
5.       foo(&i)++);
6.   }
7.   void foo(int *p){
8.       printf("%d\n", *p);
9.   }
```

B.

```
1. #include <stdio.h>
2. int main(){
3.       int i = 97, p = &i;
4.       foo(&p);
5.       printf("%d", *p);
6.       return 0;
7. }
8. void foo(int **p){
9.       int j = 2;
10.      *p = &j;
11.      printf("%d","p);
12.}
```

C.

```
1. #include <stdio.h>
2. int main(){
3.       int i = 10
4.       int *const p = &i
5.       foo(&p);
6.       printf("%d\n", *p);
7. }
8. void foo(int **p){
9.       int j = 11
10.      *p = 8j
11.      printf("%d\n", **p);
12.}
```

D.

```
1. #include <stdio.h>
2. int main(){
3.   int a = 1, b = 2, c = 3;
4.   int *ptr1 = &a, *ptr2 = &b, *ptr3 = &c;
5.   int **sptr = &ptr1; //-Ref
6.   *sptr = ptr2;
7. }
```

E.

```
1. int main() {
2.     int a = 0,b = 2,c = 1,d = 4;
3.     int f, g, h;
4.     char ch1 ='a'; //ASCII 'a'=97
5.
6.     a = (b + c) * (ch1 + d);
7.     printf("%d\n", a);
8.
9.     f = (g = a + b) >= c;
10.    printf("%d\n", f);
11.
12.    h = g == (d - 1);
13.    printf("%d\n", h);
14.
15.    ch1 = ch1 + 1;
16.    printf("%d\n", ch1);
17.    return 0;
18.}
```

F.

```
1. int func(int x) {
2.     return x % 4 + 1;
3. }
4.
5. int main() {
6.     int b = 5;
7.     int y = 2 + func(3 * b + 1);
8.     int z = func(func(y));
9.
10.    printf("%d - %d\n", y, z);
11.
12.    return 0;
13.}
```

G.

```
1. void func2();
2. void func1() {
3.     printf("In function 1\n");
4.     func2();
5. }
6. void func2() {
7.     printf("In function 2\n");
8.     func1();
9. }
10.int main() {
11.    func1();
12.    func2();
13.
14.    return 0;
15.}
```

H.

```
1. int main() {
2.     int flag = (int)(1.5 + 6) % 4;
3.     if (!flag){
4.      printf("*%d\n",(5>=6 || 1<8 && 9==7));
5.     }
6.     printf("@%d\n",(5+1==6 || 8-1>4));
7.     return 0;
8. }
```

PREDICT THE OUTPUT ANSWERS:

A. Compiler Error
B. 2 2
C. 11 11
D. ptr1 points to b
E. 303
   1
   0
   98
F. 3 - 1
G. In function 1
   In function 2
   In function 1
   (infinite calls)
H. @1

Q1) Write a C program to hold two integer pointers as structure members. Allocate space for the structure and its data members during runtime. Get one array as input. In the second array copy the elements of the first array and replace the odd positioned elements by the product of its adjacent elements. Access the array elements and structures using pointers instead of subscript notation.

First array (Input): 1 2 3 4 5 6
Second array (Output): 1 3 3 15 5 6

Q2) Write a program to create two arrays with minimum five elements each. Merge the arrays to a new array in such a way that first array may be copied as it is and reverse only the second array and merge it. Perform sorting in the new array and print it. Implement the same by passing appropriate arrays to functions. Below is the sample output.

Sample I/P and O/P

Enter the number of elements for First Array : 4
Enter the elements for First Array : 4 13 12 1
Enter the number of elements for Second Array : 4
Enter the elements for Second Array : 4 6 7 8 9

Elements After Merging 4 13 12 1 9 8 7 6
The sorted elements are 1 4 6 7 8 9 12 13

Q3) Write a program in C for the following: Get two matrices of varying dimensions. Check the dimensions satisfy the matrix multiplication criteria. Print the input matrices in matrix form. Implement a separate user defined function for multiplying matrices and this function should have recursive call feature to get the results. Print the output matrix in matrix form.

Q4) A statistician reads two sets of data for his research work. Set A = {women's height in centimeters}. Set B = {men's height in centimeters}. Each set contains five data. He is passing both the set of data to a function using the base address of the dataset A & B. Inside the function he merges all people heights and producing the result in increasing order. The ordered height should be displayed in main. Use pointer notation to access the data.

Eg. A = {34,12,56,7,25} and B = {112,23,4,77,89}
Output C = {4,7,12,23,25,34,56,77,89,112}

Q5) Write a C program for the following data: Get Loan amount from the user, Loan should be completed in 10 months" period of time, 10% of the loan amount will be collected as monthly fixed principal, 3% interest can be calculated from closing month balance. Print the data as follows:

For eg: If Loan amount =4000

Month	Monthly Fixed Principle Amount	3% Interest for remaining balance	Balance
1	400	120	3600
2	400	108	3200
3	400	96	2800
- - - -			
10	400	12	0

Q6) Develop a c program in which the user enters running text of finite length with space. Accommodate the data in a character array and pass it as reference to a function called extractor. Extractor function copies 'n' characters from mthposition of the given text. Store the extracted text and display it in the main.

Eg. Input: hello how are you?
From 3rd position (not index) onwards extract 7 characters
Output: llo how

Q7) A structure contains name of cricketer, his age, and number of test matches that he has played and the average runs that he has scored in each test match. Create an array of structure to hold records of 20 such cricketer and then write a program to read these records and arrange them in ascending order by average runs and write the sorted details in a file ('cricketer.txt')

Q8) Rohan has been given an array A of size N.Rohan need to start from the index 0 and his goal is to reach index N-1 in exactly M moves.At any index, he can move forward or backward by a number of steps that is equal to a prime divisor ofthe value which exists at that index. Rohan cannot go beyond the array while going forward or backward.
Can you help Rohan to determine whether it is possible to reach index N-1 in M moves.

Constraints:
1≤ T < 10
2≤ N <40
1≤ *A*[i] <500
1≤ M ≤50

Input Format:
First line: *T* (number of test cases)
First line in each test case: *N*
Second line in each test case: *N* space-separated integers (denoting the array *A*)
Third line in each test case: *M*

Output Format: For each test case, print YES or NO depending upon the result.

Q9) Tina has recently been introduced to a programming concept called Hashing.
Hashing is the transformation of a string of characters into a usually shorter fixed-length value or key that represents the original string.Hashing is used to index and retrieve items in a database because it is faster to find the item using the shorter hashed key than to find it using the original value. As a homework, her teacher has given a string and asked her to find no of strings she can make using each of the alphabet as starting character. Though she has already solved the question, she isn't sure if her answers are correct. Help her by telling the correct answer of all the questions.
For each alphabet, print the remainder obtained by dividing the answer by 1000000007.

Constraints:
1<=Length of string<=100000
Input Format:
Only line of Input has a single string.
Output Format:
26 integers denoting no of strings that can be made by using each of the alphabet as starting character.
Q10) Rahul who studies arts came across a programming challenge of finding the distance between the two array values is the number of indices between them. Given 'a', find the minimum distance between any pair of equal elements in the array. If no such value exists, return -1

Constraints:
1 <= n <= 10^3
1 <= a[i] <= 10^5
Input Format:
The first line contains an integer 'n', the size of array 'a'.
The second line contains 'n' space-separated integers a[i].
Output Format:
Print a single integer denoting the minimum d[i, j] in 'a'. If no such value exists, print -1

Q11) Polycarp has an array consisting of n integers. He wants to play a game with this array. The game consists of several moves. On the first move, he chooses any element and deletes it (after the first move the array contains n−1 elements). For each of the next moves he chooses any element with the only restriction: its parity should differ from the parity of the element deleted on the previous move. In other words, he alternates parities (even-odd-even-odd-... or odd-even-odd-even-...) of the removed elements. Polycarp stops if he can't make a move.
Formally:

1. If it is the first move, he chooses any element and deletes it;
2. If it is the second or any next move:
- if the last deleted element was odd, Polycarp chooses any even element and deletes it;
- if the last deleted element was even, Polycarp chooses an odd element and deletes it.
3. If after some move Polycarp cannot make a move, the game ends.

Polycarp's goal is to minimize the sum of non-deleted elements of the array after the end of the game. If Polycarp can delete the whole array, then the sum of non-deleted elements is zero. Help Polycarp find this value.

Constraints:
$1 \le n \le 2000$
$0 \le a_i \le 10^6$

Input Format:
The first line of the input contains one integer n — the number of elements of a.
The second line of the input contains n integers a1,a2,…,an, where ai is the i-th element of a.

Output Format:
Print one integer — the minimum possible sum of non-deleted elements of the array after the end of the game.

Q12) Priya got a new doll these days. It can even walk!
Priya has built a maze for the doll and wants to test it. The maze is a grid with n rows and m columns. There are k obstacles, the i -th of them is on the cell (xi,yi) , which means the cell in the intersection of the xi -th row and the yi -th column.
However, the doll is clumsy in some ways. It can only walk straight or turn right at most once in the same cell (including the start cell). It cannot get into a cell with an obstacle or get out of the maze.

More formally, there exist 4 directions, in which the doll can look:
1. The doll looks in the direction along the row from the first cell to the last. While moving looking in this direction the doll will move from the cell (x,y)into the cell (x,y+1) ;
2. The doll looks in the direction along the column from the first cell to the last. While moving looking in this direction the doll will move from the cell (x,y) into the cell (x+1,y);
3. The doll looks in the direction along the row from the last cell to first. While moving looking in this direction the doll will move from the cell (x,y) into the cell (x,y−1);
4. The doll looks in the direction along the column from the last cell to the first. While moving looking in this direction the doll will move from the cell (x,y) into the cell (x−1,y).

Standing in some cell the doll can move into the cell in the direction it looks or it can turn right once. Turning right once, the doll switches it's direction by the following rules: 1→2 , 2→3 , 3→4, 4→1 . Standing in one cell, the doll can make at most one turn right.

Now Priya is controlling the doll's moves. She puts the doll in of the cell (1,1) (the upper-left cell of the maze). Initially, the doll looks to direction 1 , so along the row from the first cell to the last. She wants to let the doll walk across all the cells without obstacles exactly once and end in any place. Can it be achieved?

Constraints:
$1 \le n, m \le 10^5, 0 \le k \le 10^5$
$1 \le xi \le n, 1 \le yi \le m$

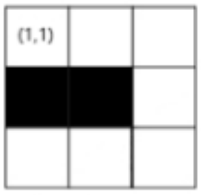
Input Format:
The first line contains three integers n , m and k, separated by spaces — the size of the maze and the number of obstacles. Next k lines describe the obstacles, the ith line contains two integer numbers xi and yi , separated by spaces, which describes the position of i th obstacle. It is guaranteed that no two obstacles are in the same cell and no obstacle is in cell (1,1).

Output Format:
Print 'Yes' (without quotes) if the doll can walk across all the cells without obstacles exactly once by the rules, described in the statement. If it is impossible to walk across the maze by these rules print 'No' (without quotes).

Explanation:
sample input
3 3 2
2 2
2 1

sample output
Yes

Note: Here is the picture of the maze described in the first example:



In the first example, the doll can walk in this way:
The doll is in cell (1,1), looks to direction 1. Move straight;
The doll is in the cell (1,2), looks to the direction 1. Move straight;
The doll is in the cell (1,3), looks to direction 1. Turn right;
The doll is in the cell (1,3), looks to direction 2. Move straight;
The doll is in the cell (2,3), looks to direction 2. Move straight;
The doll is in cell (3,3), looks to direction 2. Turn right;
The doll is in cell (3,3), looks to direction 3. Move straight;
The doll is in cell (3,2), looks to direction 3. Move straight;
The doll is in cell (3,1), looks to direction 3. The goal is achieved, all cells of the maze without obstacles passed exactly once.

Q13) Laaslya is planning to go to the cinema theater to spend her weekend vacation. Her friends Tina, Caleb, and Jocelyn all knew about Laasya's plan. They say we are coming too, but she thinks to ignore them because only Laasya has enough money to pay for the cinema ticket.

Laasya is very good at programming, so she puts up a puzzle to avoid taking her friends to the cinema. She also says that those who have finished this can come with me.
The puzzle is Tower of Hanoi is a mathematical puzzle where we have three rods and n disks. The objective of the puzzle is to move the entire stack to another rod. the number of the disk will be given as input obeying the following simple rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.
3. No disk may be placed on top of a smaller disk.

Constraints:
1≤ n ≤1000
Input Format:
The input represents the single line integer <n=
Output Format:
Display the Movement of the disk.

Q14) A deck of cards contains 52 cards in total, distributed into 4 suits namely Heart g, Diamond h, Club g and Spade f. Each suit has the following 13 cards: 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King, and Ace. A player will draw cards randomly from a deck of cards and will earn 5 points on each numbered card (i.e., cards with numbered values from 2 to 10) and 10 points on each face-value card (i.e., A, K, Q or J). The game will end if a Jack card is drawn immediately after an Ace card. In that case, the last Jack card will not contribute anything to the score.
Write a C program to simulate this game and display the total number of cards draw and points earned on game termination. Be sure to allow uppercase as well as lowercase letters as input. An ASCII chart is given below for your reference. The program should also terminate immediately if an incorrect value for card is entered by the user. Display an appropriate message to notify the user about program termination status (see sample outputs below).
Two game samples are given below for your understanding but please DO NOT HARD-CODE for either of the scenario or otherwise. You are also not allowed to use break/continue statements in this problem for any purpose. Note that in the first sample input, T represents the number card 10 (you may use any other character of your choice).

Sample Run 1: T 5 A J
Output:
Total cards draw = 4
Total points = 20
Program terminated on getting a combination of Ace and Jack cards

Sample Run 2: 4 j 6 A k B
Output:
Total cards draw = 5
Total points = 40
Program terminated on wrong input

Character	ASCII
a	97
b	98
c	99
d	100
e	101
f	102
g	103
h	104
i	105
j	106
k	107
l	108
m	109

Character	ASCII
n	110
o	111
p	112
q	113
r	114
s	115
t	116
u	117
v	118
w	119
x	120
y	121
z	122

Character	ASCII
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77

Character	ASCII
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90

Character	ASCII
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57

Q15) A video player plays a game in which the character competes in a hurdle race. Hurdles are of varying heights, and the characters have a maximum height they can jump. There is a magic potion they can take that will increase their maximum jump height by '1' unit for each dose. How many doses of the potion must the character take to be able to jump all of the hurdles. If the character can already clear all of the hurdles, return 0.

Constraints:

1<=n, k<=10

1<=height[i] <=100

Input Format:

The first line contains two space-separated integers 'n' and 'k', the number of hurdles, and the maximum height the character can jump naturally.

The second line contains 'n' space-separated integers height[i] where 0<=i<n.

Output Format:

Print the potion must the character take to be able to jump all of the hurdles. If the character can already clear all of the hurdles, return 0.

Explanation:

height = [1,2,3,3,2]

k=1

The character can jump 1 unit high initially and must take 3-1=2 doses of potion to be able to jump all of the hurdles.