Name: Section: Rollno:

ESC101: Fundamentals of Computing(Lab Exam 1)

12th February, 2014

Total Number of Pages: 6 Total Points 100

Question	Points	Score
1	30	
2	30	
3	30	
4	10	
Total:	100	

Instructions

1. Read these instructions carefully.
2. DO NOT use the `math.h` library for any of the questions.
3. For every program that you write, use the following naming convention:
   If your IITK id is **aturing**, your roll no is **13999** and you are in section **A6**, then your program name for
   Question 1 should be `a6-aturing-13999-q1.c`
   Question 2 should be `a6-aturing-13999-q2.c`
   Question 3 should be `a6-aturing-13999-q3.c`
   Question 4 should be `a6-aturing-13999-q4.c`
4. Every program should complete its execution within 5 seconds, on every input. If it takes longer time then marks will be deducted even if your answer is correct.
5. Range of `int` is -2147483648 to 2147483647.
6. Placeholder for `long int` is `%ld` and for `long long int` is `%lld`
7. Apart from the given test cases, you should check the correctness of your program on your own test cases as well.

Helpful hints

1. The questions are arranged according to the increasing order of difficulty.

Question 1. (30 points) **Number of inversions**

Given an array A of integers, if $A[i] > A[j]$ and $i < j$, then (i, j) is called an *inversion pair*.

A *k-ary inversion pair* is an inversion pair (i, j) such that $|i - j| \leq k$.

Write a program to do the following:

1. Takes integers n and k as input. (Assume that $n \leq 100$, and $0 \leq k \leq n$)
2. Then take n integers as input say $a_1, a_2, \ldots a_n$ in an array.
3. Print the number of k-ary inversion pairs in the array.
   Two k-ary inversion pairs (i_1, j_1) and (i_2, j_2) are counted different if either $i_1 \neq i_2$, or $j_1 \neq j_2$ or both.

Here are some sample interactions of the program:

```
$./a.out
Enter n: 3
Enter k: 1
Enter number: 5
Enter number: 2
Enter number: 4
Number of K-ary inversion pairs: 1

$./a.out
Enter n: 3
Enter k: 2
Enter number: 5
Enter number: 2
Enter number: 4
Number of K-ary inversion pairs: 2
```

Test Cases

Value of n	Value of k	n integers	*Expected Output*
5	2	12 15 18 20 18	1
15	5	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	0
10	10	10 9 8 7 6 5 4 3 2 1	45
8	5	-1 -2 -3 -10 8 12 15 -29	11
20	10	-193 249 -927 658 -70 272 544 -122 923 709 -560 -835 -508 42 987 -497 -673 729 -160 -388	70
20	15	-193 249 -927 658 -70 272 544 -122 923 709 -560 -835 -508 42 987 -497 -673 729 -160 -388	92
20	20	-193 249 -927 658 -70 272 544 -122 923 709 -560 -835 -508 42 987 -497 -673 729 -160 -388	97

Question 2. (30 points) **Max Power**

Write a program to do the following:

1. Take an integer n as input. (Assume that $2 \leq n \leq 10^6$)
2. Print the maximum value of an integer b such that n can be written as a^b for some integer a.

Examples:

- For $n = 5$, the maximum value of $b = 1$, because 5 can only be written as 5^1.
- For $n = 16$, the maximum value of $b = 4$, since 16 can be written as 2^4.

Here are some sample interactions of the program:

```
$./a.out
Enter n: 5
Value of b: 1

$./a.out
Enter n: 81
Value of b: 4
```

Test Cases

Value of `n`	*Expected Output*
2053	1
103823	3
131072	17
46656	6
994009	2

Question 3. (30 points) **Counting Telephone calls**

Write a program to do the following:

1. Take an integer n as input. (Assume that $n \leq 100$), which is the number of calls to read.
2. Read n lines each containing two integers s and e which is the start time and the end time for a telephone call respectively (assume $s \leq e$).
3. Take integer "q" as input. (Assume that $q \leq 100$), the number of queries to follow respectively.
4. Read q lines each containing two integers u and v which is the start time and the end time of the query interval respectively (assume $u \leq v$).
5. For each query, print the number of calls (among the "n" calls) that were active in that query interval.

Note: A call is said to be active in an interval if the call time overlaps with the interval for **at least one second**.

Here are a few sample interactions of the program:

```
$./a.out
Enter the number of calls (n): 1
Enter call 1: 0 10
Enter the number of query intervals (q): 2
Enter query interval 1: 9 10
Active calls: 1
Enter query interval 2: 11 11
Active calls: 0
```

```
$./a.out
Enter the number of calls (n): 3
Enter call 1: 2 7
Enter call 2: 0 10
Enter call 3: 5 13
Enter the number of query intervals (q): 2
Enter query interval 1: 0 6
Active calls: 3
Enter query interval 2: 7 10
Active calls: 2
```

Test Cases

Number of Calls	Start Time and End Time of the Calls	Number of Query Intervals	Start Time and End Time of the Query Interval	*Expected Output*
5	5 8 8 9 9 14 2 8 12 15	1	8 12	2
5	5 8 8 9 14 14 2 8 15 15	3	8 12 14 15 15 15	1 0 0
5	1 5 5 18 8 29 10 12 5 6	2	15 15 12 12	0 0

Question 4. (10 points) **Longest Increasing Subsequence** (*This is a tough problem*)

Given a sequence of elements $S = (a_1, a_2, \ldots, a_n)$, a *subsequence* is a sequence that is obtained from S by deleting some (or none of the) elements without changing the order of the remaining elements.

For example, consider the sequence (A, B, C, D, E, F, G):

- (A, B, D) is a subsequence.
- (B, E, F, G) is a subsequence.
- (B, D, A) is **not** a subsequence because the order of elements is changed.
- (A, B, C, D, E, F, G) is also a subsequence.
- (D, E, H) is **not** a subsequence because it contains elements not present in the original sequence.

The *longest increasing subsequence* of a sequence of numbers is a subsequence in which the subsequence's elements are in increasing order, lowest to highest, and the length of the subsequence is maximum over all such subsequences. (Note that the longest increasing subsequence is not necessarily contiguous, or unique.)

For example, consider the sequence $(0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15)$. Then, a longest increasing subsequence is $(0, 2, 6, 9, 13, 15)$.

This subsequence has length six; the input sequence has no seven-member increasing subsequences. The longest increasing subsequence in this example is not unique. For instance, $(0, 4, 6, 9, 11, 15)$ or $(0, 4, 6, 9, 13, 15)$ are other increasing subsequences of equal length in the same input sequence.

Write a program to do the following:

1. Take an integer N as input, which is the number of elements in the sequence (assume that $0 \leq N \leq 1000$).
2. Input N elements of the sequence (each of type `int`). All the numbers in the sequence are distinct.
3. Print the length of the longest increasing subsequence.

Here are some sample interactions of the program:

```
$./a.out
Enter N: 5
Enter the sequence: 1 2 3 -1 5
Length of LIS: 4

$./a.out
Enter N: 6
Enter the sequence: 100 79 67 45 31 12
Length of LIS: 1

$./a.out
Enter N: 4
Enter the sequence: -100 12 43245 142453
Length of LIS: 4
```

Test Cases

Value of `N`	Elements in the sequence	*Expected Output*
6	5 6 20 7 8 21	5
10	-10 100 -9 101 -8 102 -7 103 -6 104	6
20	-90 7894 -289 89048 89076 7893 780 7895 790 80 78907 79 -78 89065 32 543 14233 142332 4353453 67	7