A. Requirements

Code (90%)

You can write your code in Java, Python, C, or C++. The *time limit* may vary among different languages, depending on the performance of the language. Your code must be a complete excutable program instead of only a function. We guarantee test data strictly compliance with the requirements in the description, and you do not need to deal with cases where the input data is invalid.

No AI Assistance or Plagiarism: All code must be your own. The use of AI tools (e.g., ChatGPT, GitHub Copilot) or copying from external sources or peers is **strictly forbidden**.

Violations of the plagiarism rules will result in 0 points or even **failure** of this course.

Libraries in this assignment:

- For C/C++, you can only include standard library.
- For Java, you can only `import java.util.*`
- For Python, you can only import standard library. In other words, you cannot import libraries such as `numpy`.

We provide an example problem to illustrate the information above better.

Report (10%)

You also need to write a report in `pdf` type to explain the following:

- What are the possible solutions for the problem?
- How do you solve this problem?
- Why is your solution better than others?

Please note that the **maximum** number of pages allowed for your report is **5 pages**.

Remember that the report is to illustrate your thinking process. Keep in mind that your report is supposed to show your ideas and thinking process. We expect clear and precise textual descriptions in your report, and we do not recommend that you over-format your report.

B. Example Problem: A + B Problem

Description

Given 2 integers A and B, compute and print A + B

Input

Two integers in one line: A, and B

Output

One integer: A + B

Sample Input 1

```
1 2
```

Sample Output 1

```
3
```

Problem Scale & Subtasks

For 100% of the test cases, $0 \leq A, B \leq 10^6$

Solutions

Java

```java
import java.util.*;

public class Example {
    public static void main(String[] args) {
        int a, b;
        Scanner scanner = new Scanner(System.in);
        a = scanner.nextInt();
        b = scanner.nextInt();
        scanner.close();
        System.out.println(a + b);
    }
}
```

Python

```python
AB = input().split()
A, B = int(AB[0]), int(AB[1])
print(A + B)
```

C

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
  int A, B;
  scanf("%d%d", &A, &B);
  printf("%d\n", A + B);
  return 0;
}
```

C++

```cpp
#include <iostream>

int main(int argc, char *argv[])
{
  int A, B;
  std::cin>> A >> B;
  std::cout<< A + B << std::endl;
  return 0;
}
```

C. Submission

After finishing this assignment, you are required to submit your code to the Online Judge System (OJ), and upload your `.zip` package of your code files and report to BlackBoard.

C.1 Online Judge

Once you have completed one problem, you can submit your code on the page on the Online Judge platform (oj.cuhk.edu.cn, campus only) to gain marks for the code part. You can submit your solution of one problem for **no more than 80 times**.

After you have submitted your program, OJ will test your program on all test cases and give you a grade. The grade of your latest submission will be regarded as the final grade of the corresponding problem. Each problem is tested on multiple test cases of different difficulty. You will get a part of the score even if your algorithm is not the best.

Note: The program running time may vary on different machines. Please refer to the result of the online judge system. OJ will show the time and memory limits for different languages on the corresponding problem page.

If you have other questions about the online judge system, please refer to OJ wiki (campus network only). If this cannot help you, feel free to contact us.

C.2 BlackBoard

You are required to upload your **source codes and report** to the BlackBoard platform. You need to name your files according to the following rules and compress them into `A3_<Student ID>.zip` :

```
A3_<Student ID>.zip
|-- A3_P1_<Student ID>.java/py/c/cpp
|-- A3_P2_<Student ID>.java/py/c/cpp
|-- A3_Report_<Student ID>.pdf
```

For Java users, **you don't need to consider the consistency of class name and file name**.

For example, suppose your ID is 123456789, and your problem 1 is written in `Python`, problem 2 is written in `Java` then the following contents should be included in your submitted `A3_123456789.zip`:

```
A3_123456789.zip
|-- A3_P1_123456789.py
|-- A3_P2_123456789.java
|-- A3_Report_123456789.pdf
```

C.3 Late Submissions

Submissions after Nov.22 2024 23:59:00(UTC+8) would be considered as LATE.

The LATE submission page will open after deadline on OJ.

Submisson time = max{latest submisson time for every problem, BlackBoard submisson time}

There will be penalties for late submission:

- 0–24 hours after deadline: final score = your score×0.8
- 24–72 hours after deadline: final score = your score×0.5
- 72+ hours after deadline: final score = your score×0

FAQs

Q: My program passes samples on my computer, but not get AC on OJ.

A: Refer to OJ Wiki Q&A

Authors

If you have questions for the problems below, please contact:

- Problem 1. Chunxu Lin: 221012033@link.cuhk.edu.cn
- Problem 2. Diyuan Deng: diyuandeng@link.cuhk.edu.cn

CSC3100 Data Structures Fall 2024

Programming Assignment 3

Chunxu Lin: 221012033@link.cuhk.edu.cn
Diyuan Deng: diyuandeng@link.cuhk.edu.cn

Due: Nov.22 2024 23:59:00

Assignment Link: https://oj.cuhk.edu.cn/d/csc3100_2024_fall/homework/672b68fce40c360196c1f796

1 Tree (40% of this assignment)

1.1 Description

Warrior has two important attributes, **magic point**(MP) and **health point**(HP). These two values are **determined by the warrior** at the beginning.

The magic city can be regarded as an undirected simple connected **tree** (data structure) with n nodes and m edges. the warrior will walk in the city, from **any starting leaf node** to destination t which is the root node. Every time the warrior passes an edge, the magic point of the warrior will be reduced by 1. At the same time, there is a monster with attack power on each edge, and the warrior will fight with it. If the magic point of the warrior before passing this edge is k, and the attack power of the monster on this edge is w, then the battle when passing this edge will consume $\max(0, w-k)$ healthy points.

The warrior wants to ensure that his magic point and health point are **both zero** when reaching node t at this time. Please find the minimum health point that the warrior needs at **any** beginning nodes.

For example, let's take n = 2, m = 1, and t = 1, there is only one edge (1, 2) with a monster whose attack power is 9. The warrior needs at least 8 healthy points as the warrior will spend $8 = \max(0, 9-1)$ healthy points to cross this edge.

Input

```
1 2 1
1 2 9
```

Output

```
8
```

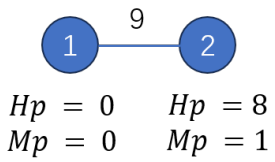


Figure 1: Example.

1.2 Input

- The first line has three integers n, m, t where n and m present the number of nodes and edges respectively.
- The next m lines, each with three integers u, v, w, means there is an edge between the nodes u and v, and the attack power of the monster on the edge is w.(u and $v = 1, 2, ..., n$)

1.3 Output

- The minimum health point that the warrior needs at the beginning.

Sample Input 1

```
8 7 1
1 2 5
2 3 5
3 4 5
4 5 5
5 6 5
6 7 5
7 8 5
```

Sample Output 1

```
10
```

You can build a new tree and find the minimal health point that the warrior needs at the leaf node.

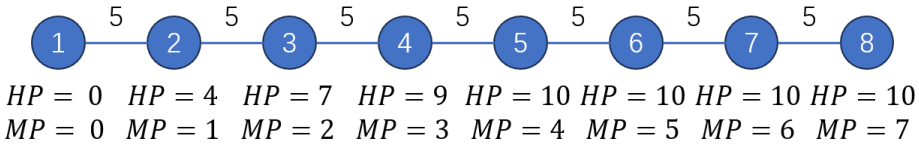


Figure 2: The new tree of Sample 1

Sample Input 2

```
8 7 1
1 2 5
2 3 1
2 4 7
1 5 4
5 6 1
6 7 2
5 8 3
```

Sample Output 2

```
9
```

Same with Sample 1, the maximal minimal health point is 8 at leaf node 4.

Sample Input 3

```
See attached q1sample.in
```

Sample Output 3

```
See attached q1sample.out
```

Problem Scale & Subtasks

- $t = 1$, $m = n - 1$, and $0 \leq w$

Test Case No.	Constraints
1-4	$2 \leq n < 10$
5-8	$10 \leq n < 100$
9-10	$100 \leq n \leq 1000$

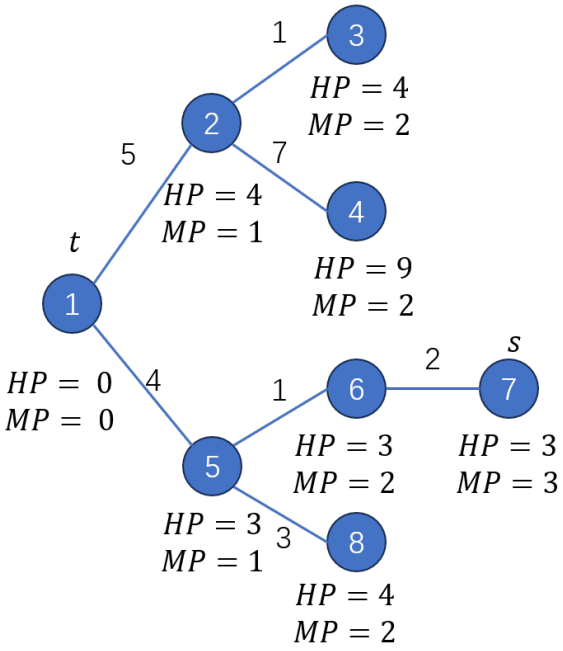


Figure 3: The new tree of Sample 2

2 Find the Maximum Subinterval Sum with Unequal Shelf Contributions (50% of this assignment)

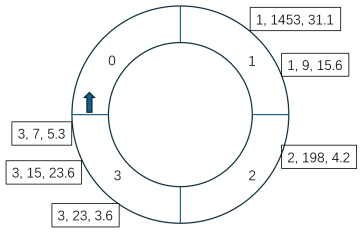
2.1 Description

You are an outstanding employee of a supermarket. There are n items in the supermarket, each with a unique ID; there are also k shelves, which form **a ring by connecting the first and last shelf.** Your bag has a limited capacity, and you can only take a series of items whose total number is smaller or equal to **bag_size**.

> Let's take n=6, k=4, bag_size=3 for example.
>
> Then 'n=6' means there are 6 pieces of items in total,
>
> 'k=4' stands for that there are 4 shelves in total and the selves' ids are 0, 1, 2, 3 respectively. The four shelves are placed in a ring, such that shelf_0 is placed is right in front of shelf_1 and right behind shelf_3.
>
> Your bag has the size of 3, so you can take things whose whole number is smaller or equal to 3.

The supermarket categorizes items using a hashing method: each item's ID determines its assigned shelf number, calculated by **id%k**. The shelves are numbered from 0 to $k-1$, and on each shelf the items are placed in **descending order** of their IDs, from front to back.

```
6  4  3
15      23.6
23      3.6
198     4.2
7       5.3
9       15.6
1453    31.1
```



Here ,you can record them in tuples (shelf_num, id, value), and you will get the following tuples:

$$(3, 15, 23.6),\ (3, 23, 3.6),\ (2, 198, 4.2),\ (3, 7, 5.3),\ (1, 9, 15.6),\ (1, 1453, 31.1)$$

After the hashing function and the sorting procedure, the final result is shown on the upper_right, we can find out which shelf each item belongs to.

Today, your boss is in a great mood and has given you the opportunity to take any **continuous** sequence of items from the shelves for free. However, there are two important restrictions:

- You are only allowed to take thing **continously**, and the **continuous** sequence of items whose total number must be smaller or equal to `bag_size`.

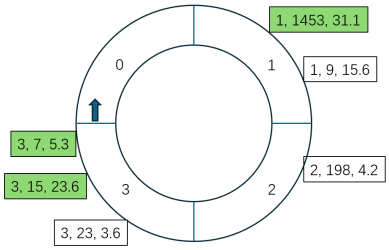
  Here you are not allowed to take (1,9,15.6), (1,1453,31.1) and (3,15,23.6) simultaneously.

- You **cannot** take an equal number of items from any two shelves. The number of items taken from each shelf must be different.

  Here you are not allowed to take (1,9,15.6), (2, 198, 4.2) and (3,23,3.6) simultaneously, because that is illegal.

- If there exists one and only one empty shelf between any two shelves with goods, we can skip it and consider the goods between the two non-empty shelves to be continuous; but if there are multiple continuous empty shelves between any two shelves with goods, we cannot connect them and consider them to be disconnected.

  Here, since there is nothing on shelf 0, we can ignore the shelf 0 and assume that (3, 7, 5.3) and (1, 1453, 31.1) are items placed next to each other and can be taken together. One optimal solution is taking 2 items from the third shelf and 1 item from the first shelf, yielding $23.6 + 5.3 + 31.1 = 60.0$.



In normal cases, what is the maximum total value of the items you can take for free? Please try to write a code to solve it.

2.2 Input

The first line contains three integers: n, k (the base for the hash operation), bag_size (representing the total number of items you can take). In the next n lines, each line contains two values:

- id[i] (a positive integer where $1 \leq i \leq n$),
- value[i] (the value of item i, a positive floating-point number where $1 \leq i \leq n$).

2.3 Output

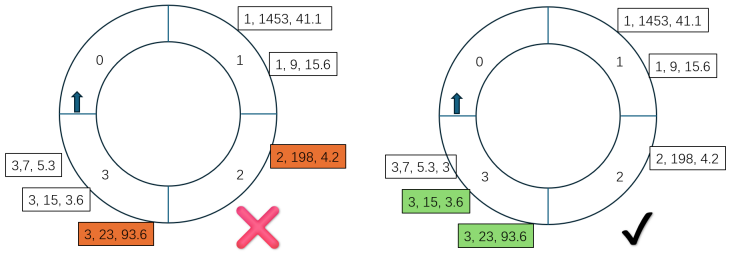
Output the maximum sum of values in a subinterval of size bag_size, rounded to one decimal place, ensuring that the number of items taken from each shelf is unequal.

Sample Input 1

```
6  4  2
15       3.6
23       93.6
198      4.2
7        5.3
9        15.6
1453     41.1
```

Sample Output 1

```
97.2
```



Sample Input 2

```
9 6 5
3289 6652.7
7803 3590.7
659 5622.2
1542 7875.3
4563 3198.4
8757 2979.0
7777 8263.7
1068 387.7
7134 4324.3
```

Sample Output 2

```
27503.7
```

Sample Input 3

```
See attached q2sample.in
```

Sample Output 3

```
See attached q2sample.out
```

2.4 Problem Scale & Subtasks

Hint: bag_size might be larger than n, k.

k> 0, id > 0, value > 0

Test Case No.	Constraints
1-3	$0 < n \leq 10$
4-5	$0 < n \leq 100$
6-7	$0 < n \leq 1000$
8-10	$0 < n \leq 5000$
11-12	$0 < n \leq 100000, bag\_size = 1$