```
a = [[1,2,3,4,5],
    [6,7,8,9,10],
    [11,12,13,14,15]]
k = 0
for i in a:
    if a.index(i) == (len(a)//2):
        for j in i:
            k += j
b = 0
c = 0
for i in a:
    for j in range(len(i)):
        mid = len(i)//2
        if j == mid:
            b += i[j]
        if a.index(i) == len(a)//2:
            if j == len(i)//2:
                c = i[j]

print(k)
print(b)
print(c)
sum = k + b -c
print(sum)
```

```
40
24
8
56
```

```
text = "ABABCDABCABCDBDCABBABABAABCD"

def find_min_time(total_nodes, total_edges, graph, traffic):

  queue = [(0, 1)] # [(time, node)]

  visited = set()

  while queue:

    queue.sort()

    time, node = queue.pop(0)

    if (time, node) in visited:

      continue
```

```
    visited.add((time, node))
    if node == total_nodes:
      return time
    for start, end, weight in graph:
      if start == node:
        new_time = time + weight
        while new_time in traffic[end]:
          new_time += 1
        queue.append((new_time, end))
total_nodes, total_edges = map(int, input().split())
graph = []
for _ in range(total_edges):
  edge = tuple(map(int, input().split()))
  graph.append(edge)
traffic = {}
for node in range(1, total_nodes+1):
  jam_times = list(map(int, input().split()[1:]))
  traffic[node] = jam_times
min_time = find_min_time(total_nodes, total_edges, graph, traffic)
print(min_time)
```

```
5
```

```
def sort_by_curse(teacher):
  return teacher[2]
total_teachers, total_days = map(int, input().split())
all_teachers = []
for _ in range(total_teachers):
```

```
  arrival, lectures, curse = map(int, input().split())
  all_teachers.append((arrival, lectures, curse))

def minimize_total_curse(total_teachers, total_days, all_teachers):

  all_teachers.sort(key=sort_by_curse, reverse=True)

  schedule = [0] * (total_days + 1)

  total_curse = 0

  for arrival, lectures, curse in all_teachers:

    for day in range(arrival, total_days + 1):

      if schedule[day] == 0 and lectures > 0:

        schedule[day] = 1

        lectures -= 1

  total_curse += lectures * curse

  return total_curse

print(minimize_total_curse(total_teachers, total_days, all_teachers))
```

```
0
```