

```
#include <assert.h>
#include <ctype.h>
#include <limits.h>
#include <math.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
char* readline();
char* ltrim(char*);
char* rtrim(char*);
char** SplitString(char*);
int parse_int(char*);
/*
```

* Complete the 'minimumAverage' function below.

*

* The function is expected to return an INTEGER

* The Function accepts

2D- INTEGER- ARRAY Customers as Parameter

*/

```
int minimumAverage(int customers_rows, int customers_columns, int** customers)
```

```
{
```

```
}
```

```
int main ()
```

```
{
```

```
FILE * fpts =
```

```
fopen(getenv("OUTPUT-PATH"), "w");
```

```
int n = parse_int(ltrim(rtrim(readline())));
```

```

int** Customers = malloc(n * Size of(int));
for (int i = 0; i < n; i++) {
    *(Customers + i) = malloc(2 * (size of(int)));
    char* Customer_Item - temp = split_string(stoln(Readline()));
    for (int j = 0; j < 2; j++) {
        int Customers_Item = parse_int(*(Customers_Item - temp + j));
        (*(Customers + i) + j) = Customers_Item;
    }
}

int result = minimum_Average(n, 2, Customers);
fprint(fptr, "%d\n", result);
fclose(fptr);
return 0;
}

char* Readline() {
    size_t alloc_length = 1024;
    size_t data_length = 0;
    char* data = malloc(alloc_length);
    while (true) {
        char* cursor = data + data_length;
        char* line = fgets(cursor, alloc_length - data_length, stdin);
        if (!line)
            break;
    }
    data_length += strlen(cursor);
    if (data_length < alloc_length - 1 || data[data_length - 1] == '\n')
        break;
    alloc_length <<= 1;
    data = realloc(data, alloc_length);
    if (!data)
        data = '\0';
    break;
}

if (data[data_length - 1] == '\n')
    data[data_length - 1] = '\0';
}

```

```

data = realloc(data, data + length);
if (!data) {
    data = '\0';
}
} else {
    data = realloc(data, data + length + 1);
    if (!data) {
        data = '\0';
    } else {
        data[data + length] = '\0';
    }
}
return data;
}

```

```

char* ltrim(char* str) {
    if (!str) {
        return '\0';
    }
    if (!*str) {
        return str;
    }
    while (*str != '\0' && isspace(*str))
        str++;
    return str;
}

```

```

char* rtrim(char* str) {
    if (!str) {
        return '\0';
    }
    if (!*str) {
        return str;
    }
    char* end = str + strlen(str) - 1;
    while (end > str && isspace(*end))
        end--;
}

```

```
* (end + 1) = '\0';
```

```
return str;
```

```
}
```

```
char** split_string(char* str) {
```

```
    char** splits = NULL;
```

```
    char* token = strtok(str, " ");
```

```
    int spaces = 0;
```

```
    while (token) {
```

```
        splits = realloc(splits, sizeof(char*) * ++spaces);
```

```
        if (!splits) {
```

```
            return splits;
```

```
        }
```

```
        splits[spaces - 1] = token;
```

```
        token = strtok(NULL, " ");
```

```
    }
```

```
    return splits;
```

```
}
```

```
int parse_int(char* str) {
```

```
    char* endptr;
```

```
    int value = strtol(str, &endptr, 10);
```

```
    if (endptr == str || *endptr != '\0')
```

```
    {
```

```
        exit(EXIT_FAILURE);
```

```
    }
```

```
    return value;
```

```
}
```



```

1 #include <assert.h>
2 #include <ctype.h>
3 #include <limits.h>
4 #include <math.h>
5 #include <stdbool.h>
6 #include <stddef.h>
7 #include <stdint.h>
8 #include <stdio.h>
9 #include <stdlib.h>
10 #include <string.h>
11
12 char* readline();
13 char* ltrim(char*);
14 char* rtrim(char*);
15 char** split_string(char*);
16
17 int parse_int(char*);
18
19 /*
20  * Complete the 'minimumAverage' function below.
21  *
22  * The function is expected to return an INTEGER.
23  * The function accepts 2D_INTEGER_ARRAY customers as
24  * parameter.
25  */
26
27 int minimumAverage(int customers_rows, int
28 customers_columns, int** customers) {
29
30 }
31
32 int main()
33 {
34     FILE* fptr = fopen(getenv("OUTPUT_PATH"), "w");
35
36     int n = parse_int(ltrim(rtrim(readline())));
37
38     int** customers = malloc(n * sizeof(int*));
39
40     for (int i = 0; i < n; i++) {
41         *(customers + i) = malloc(2 * (sizeof(int)));
42
43         char** customers_item_temp =
44 split_string(rtrim(readline()));
45
46         for (int j = 0; j < 2; j++) {
47             int customers_item =
48 parse_int(*(customers_item_temp + j));
49
50             (*(customers + i) + j) = customers_item;
51         }
52     }
53
54     int result = minimumAverage(n, 2, customers);
55
56     fprintf(fptr, "%d\n", result);

```



C
Demo.c



CODE

OUTPUT

```
49
50     int result = minimumAverage(n, 2, customers);
51
52     fprintf(fp, "%d\n", result);
53
54     fclose(fp);
55
56     return 0;
57 }
58
59 char* readline() {
60     size_t alloc_length = 1024;
61     size_t data_length = 0;
62
63     char* data = malloc(alloc_length);
64
65     while (true) {
66         char* cursor = data + data_length;
67         char* line = fgets(cursor, alloc_length -
data_length, stdin);
68
69         if (!line) {
70             break;
71         }
72
73         data_length += strlen(cursor);
74
75         if (data_length < alloc_length - 1 ||
data[data_length - 1] == '\n') {
76             break;
77         }
78
79         alloc_length <= 1;
80
81         data = realloc(data, alloc_length);
82
83         if (!data) {
84             data = '\0';
85
86             break;
87         }
88     }
89
90     if (data[data_length - 1] == '\n') {
91         data[data_length - 1] = '\0';
92
93         data = realloc(data, data_length);
94
95         if (!data) {
96             data = '\0';
97         }
98     } else {
99         data = realloc(data, data_length + 1);
100
101         if (!data) {
102             data = '\0';
103         } else {
104             data[data_length] = '\0';
```

TAB

{

}

(

)

"

&

+

RUN

```

101         if (!data) {
102             data = '\0';
103         } else {
104             data[data_length] = '\0';
105         }
106     }
107
108     return data;
109 }
110
111 char* ltrim(char* str) {
112     if (!str) {
113         return '\0';
114     }
115
116     if (!*str) {
117         return str;
118     }
119
120     while (*str != '\0' && isspace(*str)) {
121         str++;
122     }
123
124     return str;
125 }
126
127 char* rtrim(char* str) {
128     if (!str) {
129         return '\0';
130     }
131
132     if (!*str) {
133         return str;
134     }
135
136     char* end = str + strlen(str) - 1;
137
138     while (end >= str && isspace(*end)) {
139         end--;
140     }
141
142     *(end + 1) = '\0';
143
144     return str;
145 }
146
147 char** split_string(char* str) {
148     char** splits = NULL;
149     char* token = strtok(str, " ");
150
151     int spaces = 0;
152
153     while (token) {
154         splits = realloc(splits, sizeof(char*) * +
+spaces);
155
156         if (!splits) {

```

```

122     }
123
124     return str;
125 }
126
127 char* rtrim(char* str) {
128     if (!str) {
129         return '\0';
130     }
131
132     if (!*str) {
133         return str;
134     }
135
136     char* end = str + strlen(str) - 1;
137
138     while (end >= str && isspace(*end)) {
139         end--;
140     }
141
142     *(end + 1) = '\0';
143
144     return str;
145 }
146
147 char** split_string(char* str) {
148     char** splits = NULL;
149     char* token = strtok(str, " ");
150
151     int spaces = 0;
152
153     while (token) {
154         splits = realloc(splits, sizeof(char*) * +
+spaces);
155
156         if (!splits) {
157             return splits;
158         }
159
160         splits[spaces - 1] = token;
161
162         token = strtok(NULL, " ");
163     }
164
165     return splits;
166 }
167
168 int parse_int(char* str) {
169     char* endptr;
170     int value = strtol(str, &endptr, 10);
171
172     if (endptr == str || *endptr != '\0') {
173         exit(EXIT_FAILURE);
174     }
175
176     return value;
177 }

```