```
#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** split_string(char*);

int parse_int(char*);


int minimumAverage(int customers_rows, int customers_
                   columns, int** customers) {

}

int main()
{

    FILE* fptr = fopen(getenv("OUTPUT_PATH"), "w");

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

    int** customers = malloc(n * sizeof(int*));
```

```
int** customers = malloc(n * sizeof(int*));
for(int i=0; i<n; i++){
    *(customers+i) = malloc(2 * sizeof(int));
char **customers_item_temp = split_string(rtrim(
                              readline()));
for (int j=0; j<2; j++){
    int customers_item = parse_int(*(customers_
                         item_temp +j));
    *(*(customers +i)+j) = customers_item;
  }
}
int result = minimumAverage(n, 2, customers);
fprintf(fptr, "%d\n", result);
fprintf(fptr,
fclose(fptr);
return 0;
}
```

Rohit

```
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 = "";

  break;
```

```
    }
}
if (data[data_length - 1] == '1'){
    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;
}
```

pdh

```
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);
```

power

```
if (endptr == str || *endptr != '\0') {
    exit(EXIT_FAILURE);
}

return value;

}
```

Sample Input #00

```
3
0 3
1 9
2 6
```

Sample Output #00

9

Sample Input #01

```
3
0 3
1 9
2 6
```

Sample Output

8

Explanation => ~~P0 P1 P2 ordering at time t=0 as A, time=1~~ NO 1

we get the minimum average wait time is

(3+6+16)/3 = 25/3 = 8.33

Rough