```
2023041
 Ayush Kandasi
# include < assert.h>
# include < ctype in)
# include (limitsih)
# Include (mathol)
# finclude (Stdbool.h)
# Proclude (Stddefih)
# Proclude Costdintal>
-# Include (stdioth)
-# Proclude (Stalib.h)
# Prochol (String.h)
Chart readline ();
 Char + Itim (char +);
 char " rothin (char"):
 Char * Split_ String (char *);
  int parse int (chart);
  - Complete the minimum Average function below.
 * The function is expected to saturn an INTEGER
 "The function accepts
  2D- INTEGER- ARRAY Customers as Parameter
 int minimum Avenage Cont automers - rows, int automers - columns, int **
  Customers)
ent main ()
  file * Fpte =
 Fopen (geten ("OUTPUT-PATH"), "w");
```

int n = Parse - int (thrim (strim (sreadline(1)));

```
data = siculloc (data, data - length);
    If (!data) &
     data = 101;
 Felse 8
   data = secullor (data, data - length +1);
   if (!data) &
      data = 10;
   3 else $
    data (data longth J. 10);
  3
  exituen data;
  Z
 choo! I trim (chas, " Sts ) &
   1f (! str) $
    section 10%
   ŀ
  if (ixstr)2
   outworn Sta;
  while (*strt= 10'+4 (cspace (*str))
 staft;
 retwen sto;
Chart otrim (chur + Sto) &
  of (!str) &
     return 10';
 if (1+5ta) $
    return sto;
 Char * end - stot stolen (sto) - 19
while (and) = sto ff isspace ( "end)
 end --;
```

```
int " Customers malloc(no size of ( lat ));
 tor (int to o; len; it+) &
  "(austomer + 1) = malloc (2" (sies of (int)));
  char customer. Hen -temp = split string ( stolm ( seadline ( )));
 for (intj-0; j22; j++) $
    Pot customers. Hem = parse - Int ( customers - Hem-temp +1));
  (( Endtomers-+ P)+3) = Customers- Hen;
  Int presult = minimum therage (n, 2, customers);
   ffrint (Fptr, " & dln", musult);
   f Close (fpta);
   siction 0;
(har readline () $
   Size _talloc : Length = 1024;
   Size -tolata Length = 0;
  Chart data - mallor (allor-length);
  while (true ) &
   Chas + Cursor = data + doctor length;
  Chart line = facts (was or, alloc - length-douta - length, stain);
   # (the) S
     breaks
data length + = strlan (Eurson);
PF (data - length < allo c = length = 1 /1 docta [data - length - 1] == 'ha') $
     break;
 alloc-length < < = 1;
 date - realloc (data, alloc - length);
   of (Idata) &
  data = 10;
   break ;
  4
if (data [data length - 1] = = 'lo') &
  clata [data-length-1] - "10";
```

2

```
(rend + 1)= 40;
  setum Sto;
Chan ** split_stoing (chan *sto) $
  Charit & splits = NULL;
 Chart token = Stotok (Sto, " ");
  Port Spaces = 0;
  While (token) &
   Splits = dealloc (splits, size of (chart) +++ spaces);
   if (!splits) s
     outurn splits;
    spilts Espaces -1 ] = taken;
 token = Strtok (NULL," ");
 return splits;
int poorse_ int (char* str) &
    Chan* end ptr;
  Prot value = Strtol (str, fend pts, 10);
  of (end gtr = = stoll *end ptr ; = 1/01)
  exit (EXIT _ FAILURE);
  suturn value;
```

```
2
```

```
int " Customers= malloc(n " Size of (int "));
tor (int ?= 0; Kn; i++) {
 *( costomer + i) = malloc (2 * (size of (int)));
 Chart customer_ Hem -temp = Split istring (stoim (swadline ()));
 for (intj-0; j22; j++) &
   Pot customers_ item = parse_int (*(ustomers _item-temp +));
 (( Eustoner + P)+3) = Customers - Ptem;
 int ovesult = minimum-Average (n, 2, customers);
  f Print ( Fptr, " & dln", result);
   f Close (fpto);
  sutrum 0;
Charteredline () &
  Size-tulloc-length = 1024;
   Size-tdata-length=0;
 chart data - mallor (allor-length);
  while (true) $
  Char & Cursor = data + data-length;
  Chart line = facts (weres, alloc-length-data-length, stain);
   1f (! line) §
     break!
data-length + = strlen (Gurson);
PF (data - length calloc: length -1 / data [data - length -1] == '\n') $
     break;
 alloc-length <<= 1;
  date = sualloc (data, alloc - length);
   of (Idata):
  data = 1/01;
    break;
  ž
if (data [data length-1]== "\n") &
  data [data-length-1] = 110;
```

```
3
```

```
data = swalloc (data, data-length);
  If (!data) &
    data = 1/01;
Jele &
 data = suallor (data, data - length +1);
  :5- (!data) $
     data = 110;
  3 else &
   data (data length J = 1/01;
 return data;
 Chan I Itaim (Chan & Sto) &
   if (1 sta) $
    setter 10'3
  if (! * str) s
   outurn Sto:
  while (*str = 10'ff (sspace (*str))
  stoff;
  retwen Sto;
 Chart otrim (chart Sto) &
   if (!sta) $
      return' 10';
   if (i+sta) $
     return sto;
    Char * end = stot stolen (sto) - 1;
   while (and) = sto ff isspace (*end)
    end -- ;
```

```
#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>
0
1
    char* readline();
2
    char* ltrim(char*);
3
    char* rtrim(char*);
4
    char** split_string(char*);
5
6
    int parse_int(char*);
7
8
9
    * Complete the 'minimumAverage' function below.
0
1
    * The function is expected to return an INTEGER.
2
    * The function accepts 2D_INTEGER_ARRAY customers as
3
   parameter.
   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*));
       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);
```

1

5

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

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

TUT

II (!uata)

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