DATE : / 20
PAGE No

NAME → Ashish Rawat
Roll No → 2023040
CAMPUS → Dehradun
St ID → 20051025
Sub → O.S Practical

Code →

Q1.)

```
# 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*);
/*
** complete the 'minimumAverage' function
   below.
* The function is expected to return an
  Integer.
```

DATE : / 20
PAGE No

```
* The function accepts 2D_Integer_ARRAY
 customers as parameter.
*/
int minimum Averge (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~~ customers
        _ item;
   }
  }
     int result = minimumAverage (n, 2,
   Customers);
  fprint (fptr, "%d \n", result);
```

Scanned by TapScanner

DATE : / 20
PAGE No

```
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';
   }
   while (*str != '\0' && isspace (*str)){
   }
   return str;
 }
```

DATE : / 20
PAGE No

```
    fclose (fptr);
     return ();
   }
  char* readline ()&
  size-t alloc-length = 1024;
   size_t data-length = 0;
    Char * data = malloc(alloc-length);
     while (true) {
     char * data = malloc (alloc-length);
     while (true) {
     char * cursor. = data + data_length;
    char* line = fgets (cursor, alloc_length -
    data-lengh, 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;
      }
   }
```

Scanned by TapScanner

DATE : / 20
PAGE No

```
char** split_string (char* str){
char** splits = NULL;
char* token = strtok (str, " ");
int spaces = 0;
while (token){
splits = realloc (splits, size of (char*) *
 ++spaces );
if (!splits ){
   return splits;
}
 splits[spaces - 1] = token;
 ~~return~~ 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;
}
```

Each process P1 P2 P3 P4 with arrival time and cpu time respectively (0, 10) (0, 2) (0 1) (0 4).