

NAME = Ayush Rawat

University

Roll No = 1921045

Student Id = 19042075

Class Roll No = 14

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
int bt[20], p[20], wt[20], tat[20], i, n, total=0, pos,  
temp;
```

```
float avg_wt, avg_tat;
```

```
printf("Enter number of processes: ");
```

```
scanf("%d", &n);
```

```
printf("Enter Burst Time: ");
```

```
for (i=0; i<n; i++)  
{
```

```
printf("P%d: ", i+1);
```

```
scanf("%d", &bt[i]);
```

```
p[i] = i+1;
```

```
}
```

for ($i = 0; i < n; i++$)

{

 pos = i ;

 for ($j = i + 1; j < n; j++$)

 if ($bt[j] < bt[pos]$)

 pos = j ;

 }

 temp = $bt[i]$;

~~bt[i] = bt[pos];~~

$bt[pos] = temp$.

 temp = $p[i]$;

$p[i] = p[pos]$;

$p[pos] = temp$;

$(H[i \rightarrow pos] : 0 \rightarrow i) \rightarrow 0$

 wt[0] = 0;

 for ($i = 1; i < n; i++$)

 {

 wt[i] = 0;

 for ($j = 0; j < i; j++$)

$\{$
 $wt[i] = 0;$
 for ($j = 0; j < i; j++$)
 $wt[i] += bt[j];$
 $\cdot total += wt[i];$
 $\}$

$$avg-wt = (float) total / n;$$

$$total = 0;$$

~~Pointf ("n Processes Burst Time) + Waiting Time
Turnaround Time");~~
~~(bt, tpt, turn)~~

for ($i = 0; i < n; i++$)

$\{$
 $tat[i] = bt[i] + wt[i];$

$$+ \text{total} += tat[i];$$

~~Pointf ("n processes waiting time", bt, wt);~~

~~Pointf ("n processes turnaround time", bt, tat);~~
~~(bt, turn, tat)~~
 $\cdot (wt[i], tat[i])\}$

$\}$

$$avg-tat = (float) total / n;$$

$$total = 0;$$

$$\text{avg-wt} = (\text{flog} +) \text{total } / n;$$

$$\text{total} = 0;$$

printf ("n Processes Burst Time + Waiting

Time + Turnaround Time") ;

for (i = 0; i < n; i++)

{

$$(\text{total} + \text{bt}[i] + \text{wt}[i]) = \text{tot} - \text{avg}$$

$$\text{tat}[i] = \text{bt}[i] + \text{wt}[i];$$

$$\text{total} + \text{tat}[i];$$

printf ("n %d %d %d %d %d", p[i], bt[i],
wt[i], tat[i]);

}

$$\text{avg-tat} = (\text{flog} +) \text{total } / n;$$

printf ("n Average Waiting Time = %f", avg-wt);

printf ("n Average Turnaround Time = %f", avg-tat);

}

$$(\text{total} + \text{wt}[i])$$

$$(\text{total} + \text{wt}[i]) = \text{tot} - \text{avg}$$

$$(\text{total} + \text{wt}[i])$$



```
Enter number of process:4
```

```
nEnter Burst Time:np1:10
```

```
p2:9
```

```
p3:1
```

```
p4:4
```

```
nProcesses Burst Time tWaiting Time
```

```
tTurnaround Timenp3tt 1tt 0ttt1np4
```

```
tt 4tt 1ttt5np2tt 9tt 5ttt14np1tt
```

```
10tt 14ttt24nnAverage Waiting Time
```

```
=5.000000nAverage Turnaround Time=
```

```
11.000000n
```

```
...Program finished with exit code
```

NAME = Ayush Rawat
University

(JNPM) 101

Roll No. 1921045

3

(Student Id = 0919042075) roll no = 6191-33117

Class Roll No = 14

Q1 #include <assert.h> // solution = 86.00 marks

ANS # include <ctype.h>

include <limits.h> // (n > i + m) = 60

include <math.h> // (l + r) / 2 = (l + r) / 2

include <stdbool.h>

include <stddef.h> // solution = 100

include <stdint.h>

include <stdlib.h> // (l + r) / 2

include <stdlib.h> // (l + r) / 2

include <string.h>

char * readLine();

char * trim (char *);

char * trim (char *);

char ** splitString (char *);

int parseInt (char *);

int minimumAverage (int customers_200s, int o) {
 return (customers_200s * o) / customers_200s;

cout << "Customer Average = " << minimumAverage << endl;

int main()

3

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

int n = parse_int(trim(trim(readline(c))));

`int *Customer = malloc (n * sizeof (int *))`

```
for (int i = 0; i < n; i++) {  
    // code  
}
```

* $(customers + i) = \text{malloc}(\{ \text{char}^{100} \gt 3 \text{ byte} \} \text{ for } \{ \text{size of (int)} \})$.

Chart** Customer-item-temp = Split-String

• `(strim (deadline(1)))`

```
for (int j=0; j<(l;j++)){ cout << i; }
```

int customers-item = parseint(*

(customers-item-temp[i]));

$$((customers + i) * j) = customers.item;$$

3. $(\#(2n))$ Period 2 - $\#(q_2)$ $\#(2n)$

3
{cont'd.}

```
int result = minimumAverage(n, 2, customers);
```

```
fprintf(fp, "%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){

(alloc_length, pfb) = fgets(data + data_length, pfb)

char * cursor = data + data_length;

char * line = fgets(cursor, alloc_length - data_length, stdin);

if (!line){

(alloc_length, pfb) = fgets(data + data_length, pfb)

data_length += strlen(cursor);

if (data_length < alloc_length - 1 || data

[data_length - 1] == '\n') {

data = realloc(data, alloc + length);

break;

alloc_length <= 1;

data = realloc(data, alloc + length);



REDMI NOTE 8
AI QUAD CAMERA

```

if (!data) {
    data = "\0";
    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_length] != '\0') {
                data[data_length] = '\0';
            }
        }
    }
    return data;
}

```



REDMI NOTE 8 (char * ltrim (char * str)) {

QUAD CAMERA

if (!str) {

 return '\0';

}

if (!*str) {

 return str;

}

if ((str == NULL) || (*str == '\0') && isspace(*str)) {

 while ((*str != '\0') && !isspace(*str))

 (str++) ;

}

 if (str == NULL) return '\0';

 return str; }

char * trim(char * str) {

 if (!str) {

 return '\0';

 int start = [1 - strlen(str)] str++;

 if (!(*str)) {

 return str;

 }

 char * end = str + strlen(str) - 1;

 while (end >= str && !isspace(*end)) {

 end--;



REDMI NOTE 8

AI QUAD CAMERA

16MP + 8MP + 2MP + 2MP

3

* (emd + 1) = '16'

return str;

3

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; }

3

splits [spaces - 1] = token;

token = strtok (NULL, " ");

3

return splits;

3

int parse_int (char * str) {

char * endptr;

int value;

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



REDMI NOTE 8

AI QUAD CAMERA

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

}

```
return value;
```

}



REDMI NOTE 8
AI QUAD CAMERA

```
5
1 4
1 9
3
2 4
3
4
9
...Program finished with exit code
```