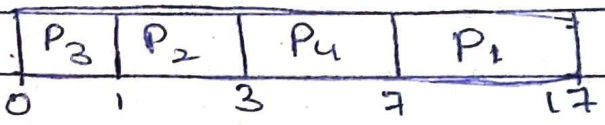
Date
Page

Ans 2:

Process	Arrival time	Burst time
P_1	0	10
P_2	0	2
P_3	0	1
P_4	0	4

Sum around time =?, waiting time =?, avg T.T =?, avg W.T =?

Sum around time = CT - AT (CT = completion time)
WT = TAT - BT



Process	AT	BT	CT	TAT	WT
P_1	0	10	17	17	7
P_2	0	2	3	3	1
P_3	0	1	1	1	0
P_4	0	4	7	7	3

avg TAT = $\frac{28}{4}$ 7// ans

avg WT = $\frac{11}{4}$ 2.75// ans.

```c
#include<stdio.h>
 int main()
{
    int
bt[20],p[20],wt[20],tat[20],i,j,n=4,to
tal=0,pos,temp;
    float avg_wt,avg_tat;
    printf("\nEnter Burst Time:\n");
    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;
    }
    wt[0]=0;
    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]+=bt[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;
    }
    wt[0]=0;
    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]+=bt[j];
        total+=wt[i];
    }
    avg_wt=(float)total/n;
    total=0;
    printf("\nPro \t  BT  \tWT  \tTAT");
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];
        total+=tat[i];
        printf("\np%d   \t %d   \t %d  \t %d",p[i],bt[i],wt[i],tat[i]);
    }
    avg_tat=(float)total/n;
    printf("\n\nAverage Waiting Time=%f",avg_wt);
    printf("\nAverage Turnaround Time=%f\n",avg_tat);
}
```

```
 Enter Burst Time:
 p1:10
 p2:2
 p3:1
 p4:4

 Pro         BT     WT         TAT
 p3          1       0          1
 p2          2       1          3
 p4          4       3          7
 p1          10      7          17

Average Waiting Time=2.750000
Average Turnaround Time=7.000000

[Process completed - press Enter]
```

algorithm

```
#include <stdio.h>
int main()
{
    int bt[20], p[20], wt[20], tat[20], i, j, n, total=0,
    pos, temp;
    float avg_wt, avg_tat;
    printf("Enter number of process:");
    scanf("%d", &n);
    printf("\nEnter Burst Time:\n");
    for(i=0; i<n; i++)
    {
        printf("p%d:", i+1);
        scanf("%d", &bt[i]);
        p[i] = i+1;
    }
```

```
    // sorting of burst times
    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;
    }

    wt[0] = 0;

    for(i=1; i<n; i++)
    {
        wt[i] = 0;
```

```
    p[i] = p[pos];
    p[pos] = temp;
  }

    wt[

  for(i=1; i<n; i++)
  {
    wt[i] = 0;
    for(j=0; j<i; j++)
    wt[i] += bt[j];

    total += wt[i];
  }
```

```
  avg_wt = (float) total/n;
  total = 0;
  printf("\nProcesst Burst Time \tWaiting Time\t
        Turnaround Time");

  for(i=0; i<n; i++)
  {
    tat[i] = bt[i] + wt[i];
    total += tat[i];
    printf("\np %d \t %d \t\t %d \t\t %d",
          p[i], bt[i], wt[i], tat[i]);
    }

  avg_tat = (float) total/n;

  printf("\n\nAverage Waiting Time = %f", avg_wt);
  printf("\n\nAverage Turnaround Time = %f",
        avg_tat);

  return 0;

}
```

Ans.

```
#include <stdio.h>
#include <ctype.h>
#include <limits.h>
#include <math.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdin.h>
#include <stdlib.h>
#include <string.h>
char* readline();
char* ltrim(char*);
char* rtrim(char*);
int parse_int(char*);
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);
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)
```

```
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; }
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'; }
```

```
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) {
```

```
*(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;
}.
```