DC126732

Huang Yanzhen

CISC2002 Lab5

|  |
| --- |
| Question 1 |
| . . .  wait\_time[i] = current\_time - proc[i].arrive;  current\_time += proc[i].service;  turnaround\_time[i] = wait\_time[i] + proc[i].service;  . . . |
|  |

|  |
| --- |
| Question 2 (1) Before commenting |
| . . .  if (p[j].arrival\_time <= current\_time && p[j].service\_time > 0)  . . .  p[shortest\_index].wait\_time = current\_time - p[shortest\_index].arrival\_time;  current\_time += p[shortest\_index].service\_time;  p[shortest\_index].turnaround\_time = current\_time - p[shortest\_index].arrival\_time;  . . . |
|  |

|  |
| --- |
| Question 2 (2) After Commenting |
| (Other code is same as above)  Commented Code:  *// if (shortest\_index == -1)*  *// {*  *// current\_time++;*  *// i--;*  *// continue;*  *// }* |
| Text  Description automatically generated |
| **What happens?**  A “bus error” occurs. It means that the process fails to attempt the required memory access.  **Why?**  The execution order of process should be: (0,1), (2,2) and (2,3) according to the principle of SJF.  However, after the execution of (0,1), *current\_time* = 1. There is no process with arrival time smaller than *current\_time*. Hence *shortest\_index* = -1.  The commented code is written to solve this problem that, as long as there’s no process with arrival time smaller than *current\_time*, *current\_time* will increase until there’s a process with a required arrivaltime without counting the process.  With this code removes, the code would directly access *p[shortest\_index]* with *shortest\_index* = -1, resulting an index-out-of-bound error of the array *p[n]*. |

|  |  |  |
| --- | --- | --- |
| Question 3 | | |
| Input (at and st) | Avg.Wait\_time(SJF) | AVG.Wait\_time(PSJF) |
| [0 1, 2 7, 3 4] | 2 | 1.33 |
| [0 2, 1 7, 3 5] | 2.33 | 2 |
| [0 5, 1 8, 5 2] | 2 | 2 |
| Graphical user interface  Description automatically generated | | |
| Graphical user interface  Description automatically generated with low confidence | | |
|  | | |