OS PRACTICAL EXAM

U18CO021- SAHIL BONDRE

U18CO021

Consider three processes, all arriving at the time Zero, with total execution time of 10, 20, 30 units, respectively. Each process spends the first 20% of execution time doing I/O, the next 70% of time doing computation and the last 10% of the time doing I/O again. The operating system uses a shortest remaining compute time first scheduling algorithm and schedules a new process either when the running process get blocked on I/O or when the running process finishes its compute burst. Assume that all I/O operations can be overlapped as much as possible. For what percentage of the does the CPU remains Idle.

```cpp
#include <stdio.h>


#include <iostream>
#include <vector>


using namespace std;


int main(int argc, char const* argv[]) {
 // process - IO, CPU, IO, P. Number
 vector<int> p1 = {2, 7, 1, 1};
 vector<int> p2 = {4, 14, 2, 2};
 vector<int> p3 = {6, 21, 3, 3};


 vector<vector<int>> ready_queue;
 ready_queue.push_back(p1);
 ready_queue.push_back(p2);
 ready_queue.push_back(p3);
```

```
int time = 0;
bool cpu_busy = false;
int cpu_busy_till = 0;
int executing_process = -1;
int cpu_time = 0;
int idle_time = 0;

// process io
while (true) {
  cout << "\n\x1B[1;32mTime is " << time << "\033[0m\n";
  for (vector<int>& p : ready_queue) {
    // process io
    if (p[0] != 0) {
      cout << "Process No." << p[3] << " is doing first IO"
           << "\n";
      p[0]--;
    }
  }

  for (vector<int>& p : ready_queue) {
    // process io
    if (p[0] == 0 && p[1] == 0 && p[2] != 0) {
      cout << "Process No." << p[3] << " is doing second IO"
           << "\n";
      p[2]--;
    }
```

```
    }


    // process cpu
    if (cpu_busy) {
      cout << "CPU is executing Process " << executing_process + 1 <<
"\n";
      cpu_time++;
      ready_queue[executing_process][1] -= 1;
      if (ready_queue[executing_process][1] == 0) {
        // process ended
        cpu_busy = false;
      }
    } else {
      // find next process to execute
      int next_process_id = -1;

      for (int i = 0; i < ready_queue.size(); ++i) {
        if (ready_queue[i][0] == 0 && ready_queue[i][1] != 0) {
          if (next_process_id == -1 ||
              ready_queue[i][1] < ready_queue[next_process_id][1]) {
            next_process_id = i;
          }
        }
      }

      if (next_process_id == -1) {
        // no process
```

```
      cpu_busy = false;
    } else {
      cpu_busy = true;
      executing_process = next_process_id;
      cpu_time++;
      ready_queue[next_process_id][1] -= 1;
      if (ready_queue[next_process_id][1] == 0) {
        // process ended
        cpu_busy = false;
      }
    }
  }
  ++time;

  // if all process done => break
  bool all_done = true;
  for (vector<int>& p : ready_queue) {
    if (p[0] != 0 || p[1] != 0 || p[2] != 0) {
      all_done = false;
      break;
    }
  }

  if (all_done) break;
}
++time;
idle_time = time - cpu_time;
```

```
 cout << "\n\x1B[1;33mTotal CPU Time: " << cpu_time << "\033[0m\n";

 cout << "\x1B[1;33mTotal Idle Time: " << idle_time << "\033[0m\n";

 cout << "\x1B[1;33mTotal Time: " << time << "\033[0m\n";

 double cpu_idle_percent = (double)idle_time * 100 / (double)time;

 printf("Percent Idle Time: %2.2f\n", cpu_idle_percent);


 return 0;

}
```

```
Total CPU Time: 42
Total Idle Time: 5
Total Time: 47
Percent Idle Time: 10.64
```