#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <limits.h>

#include <time.h>

typedef struct information

{

char user[50];

char process;

int arrival;

int duration;

int total;

} information;

information \*init(char \* userInput, char processInput, int arrivalInput, int durationInput);

void cpuScheduling(information \* list);

information \* preemption(information \*list, int time);

int freeList(information \* list);

void printOutput(information \* list);

int main(){

char \*trashCollector = NULL;

// this a pointer to the headers which would be taking into account but not used.

information \*processList;

if((processList = malloc(4\*sizeof(information)))==NULL){

return 1;

}

int d;

for(d=0; d<5; d++){

scanf("%s\t", trashCollector);

}

int i;

for (i = 0; i < 4; i++){

char expectedName[50] = "/0";

char expectedProcess = 'a';

int expectedArrival = 0;

int expectedDuration = 0;

if(scanf("%s\t%c\t%d\t%d",expectedName, &expectedProcess, &expectedArrival, &expectedDuration)){

printf("Error in scanning the data");

return 1;

}

// Create an initialize function to help store the iincoming data in the structs

information \*newStorage;

newStorage = init(expectedName, expectedProcess, expectedArrival, expectedDuration);

processList[i] = \*newStorage;

}

cpuScheduling(processList);

freeList(processList);

return 0;

}

information \*init(char \* userInput, char processInput, int arrivalInput, int durationInput){

information \* current;

if((current = malloc(sizeof(information))) == NULL)

{

return NULL;

}

strcpy(current->user, userInput);

current->process = processInput;

current->arrival = arrivalInput;

current->duration = durationInput;

return current;

}

void cpuScheduling(information \*list){

int checklist[4];

int trialTime = 0;

int count = 0;

int safteycount = 0;

int f;

for(f=0; f<4; f++){

checklist[f] = 1;

}

printf("This Would Result in:\n\tTime\tJob\n");

while(count !=4 && safteycount != 100){

information \* temp;

temp = preemption(list, trialTime);

list = temp;

count = 0;

//empty check

int e;

for(e=0; e<4; e++){

if(list[e].duration == 0 && checklist[e]!=0){

checklist[e] = 0;

}

}

int x;

for(x=0; x<4; x++){

if(checklist[x] == 0){

count++;

}

}

trialTime++;

safteycount++;

}

printf("\t%d\tIDLE\n", trialTime);

printf("\n\tSummary\n");

int spot[4];

int sumCount = 0;

int u;

for(u=0; u<4; u++){

spot[u]=0;

}

int h;

int l;

for(h=0; h<4; h++){

for(l=0;l<4;l++){

if(spot[h] < 1 && spot[l] < 1){

if(h != l && strcmp(list[h].user, list[l].user)==0){

spot[l] = 1;

spot[h] = 2;

sumCount++;

if(list[l].total > list[h].total){

list[h].total = list[l].total;

}

}

}

}

}

int value = 4-sumCount;

information sumList[value];

int increment = 0;

int o;

for(o=0; o<4; o++){

if(spot[o]!=1){

sumList[increment] = list[o];

increment++;

}

}

int p;

for(p=0;p<value;p++){

printf("\t%s\t%d\n", sumList[p].user, sumList[p].total+1);

}

}

information \* preemption(information \*list, int time){

int minDuration = 999;

int processLoc;

int check = 0;

int j;

for(j=0;j<4;j++){

if(list[j].arrival <= time){

if(list[j].duration < minDuration && list[j].duration != 0){

minDuration = list[j].duration;

processLoc = j;

check++;

}

}

}

if(check != 0){

list[processLoc].duration = list[processLoc].duration - 1;

list[processLoc].total = time;

printf("\t%d\t%c\n", time, list[processLoc].process);

}

return list;

}

int freeList(information \*list){

free(list);

return 1;

}

void printOutput(information \* in){

int j;

for(j=0;j<4;j++){

printf("%s\t%c\t%d\t%d\n", in[j].user, in[j].process, in[j].arrival, in[j].duration);

}

}