```c
#include "main.h"
#include "utils.c"


// Process pour un tour
//renvoie le temps du tour
double calculTour(struct Voiture *v){
  double tempsPit, tour = 0.0, section;
  if (out() == 1) {
    v->out = 1;
  }
  else{
    for(int i = 0; i < 3; i++){
      section = calculSection();
      if(i == 2){
        tempsPit = stand();
        if (tempsPit > 0) {
          v->pit += 1;
          section += tempsPit;
        }
      }
      tour += section;
      msleep(section);
      v->sections[i] = section;
      v->nbTour += 1;
      v->best[i] = meilleurTemps(section, v->best[i]);
    }
    v->best[3] = meilleurTemps(tour, v->best[3]);
    v->tourActuel= tour;
  }
  return tour;
}

void courseFinale(struct MemoirePartagee *tabVoitures, int i){
  struct Voiture v = tabVoitures->tableauV[i];
  struct Voiture *pointeurV = &v;
  initVarVoiture(pointeurV);


  v.nbTour= 0;

  sleep(1);

  while (v.nbTour < tabVoitures->nbTourAFaire && v.out == 0) { //tabVoitures->nbTourAFaire

    calculTour(pointeurV);
    tabVoitures->tableauV[i] = v;
    msleep(1000);
  }
  tabVoitures->nbVoituresFini += 1;
  exit(0);
}

void essaisEtQualifications(struct MemoirePartagee *tabVoitures, int i, double tempsTotal){
  struct Voiture v;
  struct Voiture *pointeurV = &v;
  if(tempsTotal >= 1080){ // soit essais soit qualif 1
    v = init_voiture(i);
  }
  else{
    v = tabVoitures->tableauV[i];
    initVarVoiture(pointeurV);

  }
  double tempsEnCours = 0;
  //Boucle course
  while (tempsEnCours < tempsTotal && v.out == 0) {
    tempsEnCours += calculTour(pointeurV);
    tabVoitures->tableauV[i] = v;
    if (tempsTotal == 5400 || tempsTotal == 3600) {
 sleep(1.5);
    }
    else{
 sleep(1.5);
    }
  }
  tabVoitures->nbVoituresFini += 1;
  exit(0);
}
```

```c
void afficheTab(struct MemoirePartagee *tabVoitures, int choix){
  struct MemoirePartagee mem = *tabVoitures;
  struct MemoirePartagee *pointeurMem = &mem;
  int finale = 0;
  if(choix == 2){
    triVoiture(pointeurMem->tableauV, 5);
    finale = 1;
  }
  else{
    triVoiture(pointeurMem->tableauV, 3);
  }
  system("clear");
  switch (choix) {
    case 0:
    printf("_____________________________________________________________________________________________\n");
    printf("|                              ESSAIS 1                              |\n");
    break;
    case 3:
    printf("_____________________________________________________________________________________________\n");
    printf("|                              ESSAIS 2                              |\n");
    break;
    case 1:
    printf("_____________________________________________________________________________________________\n");
    printf("|                              ESSAIS 3                              |\n");
    break;
    case 20:
    printf("_____________________________________________________________________________________________\n");
    printf("|                            QUALIFICATIONS 1                        |\n");
    break;
    case 15:
    printf("_____________________________________________________________________________________________\n");
    printf("|                            QUALIFICATIONS 2                        |\n");
    break;
    case 10:
    printf("_____________________________________________________________________________________________\n");
    printf("|                            QUALIFICATIONS 3                        |\n");
    break;
    case 2:

printf("_____________________________________________________________________________________________\n");

    printf("|                               COURSE FINALE                        |\n");
    break;
  }
  if (choix < 4) {
    choix = 20;
  }

  if(finale == 1){ //------------AFFICHE LA FINALE-------------------------

printf("|_____________________________________________________________________________________________|\n");

    printf("| Voiture  | Best Tour   | Tour Actuel     |   Section 1  |   Section 2  |   Section 3  | Pit  | Out  | Nb de tour|\n");
    printf("|----------|-------------|-----------------|--------------|--------------|--------------|------|------|-----------|\n");
    for(int i = 0; i < choix; i++){
        printf("|   %2d     | %.3f\"   |     %.3f\"   |   %.3f\"   |   %.3f\"   |   %.3f\"    |   %d   | %d    |   %2.0f      |\n", mem.tableauV[i].numero,
mem.tableauV[i].best[3], mem.tableauV[i].tourActuel, mem.tableauV[i].sections[0], mem.tableauV[i].sections[1], mem.tableauV[i].sections[2],
mem.tableauV[i].pit, mem.tableauV[i].out, mem.tableauV[i].nbTour);
        printf("|----------|-------------|-----------------|--------------|--------------|--------------|------|------|-----------|\n");
    }
  }
  else{ //----------------------AFFICHE LES QUALS/ESSAIS--------------
 printf("|_____________________________________________________________________________|\n");
    printf("| Voiture  | Best Tour   | Tour Actuel     |   Section 1  |   Section 2  |   Section 3  | Pit  | Out  |\n");
    printf("|----------|-------------|-----------------|--------------|--------------|--------------|------|------|\n");
    for(int i = 0; i < choix; i++){
        printf("|   %2d     | %.3f\"   |    %3.3f\"     |    %2.3f\"  |    %2.3f\"  |    %2.3f\"   |   %d   |   %d   |\n", mem.tableauV[i].numero,mem.tableauV[i].best[3],
mem.tableauV[i].tourActuel, mem.tableauV[i].sections[0],mem.tableauV[i].sections[1], mem.tableauV[i].sections[2], mem.tableauV[i].pit,
mem.tableauV[i].out);
        printf("|----------|-------------|-----------------|--------------|--------------|--------------|------|------|\n");
    }
  }
  printf("_____________________________________________________________________________\n");
  printf("|                                 Best                                      |\n");
  printf("|___________________________________________________________________________|\n");

  printf("|  Best    | Section 1    |\n");
  printf("|----------|--------------|\n");
```

```
  triVoiture(pointeurMem->tableauV, 0);
  printf("|   %2d     |  %.3f\"     |\n", mem.tableauV[0].numero, mem.tableauV[0].best[0]);
  printf("|----------|--------------|\n");
  printf("|  Best    | Section 2    |\n");
  printf("|----------|--------------|\n");
  triVoiture(pointeurMem->tableauV, 1);
  printf("|   %2d     |  %.3f\"     |\n", mem.tableauV[0].numero, mem.tableauV[0].best[1]);
  printf("|----------|--------------|\n");
  printf("|  Best    | Section 3    |\n");
  printf("|----------|--------------|\n");
  triVoiture(pointeurMem->tableauV, 2);
  printf("|   %2d     |  %.3f\"     |\n", mem.tableauV[0].numero, mem.tableauV[0].best[2]);
  printf("|----------|--------------|\n");
}

void triQualifications(struct MemoirePartagee *tabVoitures, int choix){
  triVoiture(tabVoitures->tableauV, 3); //tri de sorte que les voitures qui ont finies restent à leur place dans le tableau
  switch (choix) {
    case 20:
    for (int i = 15; i < 20; i++) {
      tabVoitures->courseFinale[i] = tabVoitures->tableauV[i];
      initVarVoiture(&tabVoitures->tableauV[i]);
      tabVoitures->tableauV[i].best[3] = i + 1000;
    }
    break;
    case 15:
    for (int i = 10; i < 15; i++) {
      tabVoitures->courseFinale[i] = tabVoitures->tableauV[i];
      initVarVoiture(&tabVoitures->tableauV[i]);
      tabVoitures->tableauV[i].best[3] = i + 1000;
    }
    break;
    case 10:
    for (int i = 0; i < 10; i++) {
      tabVoitures->courseFinale[i] = tabVoitures->tableauV[i];
      initVarVoiture(&tabVoitures->tableauV[i]);
      tabVoitures->tableauV[i].best[3] = i + 1000;
    }
    break;
  }

}

void fonctionPere(struct MemoirePartagee *tabVoitures, int choix){
  int calculNbFini;
  if(choix > 4){
    calculNbFini = choix;
  }
  else{
    calculNbFini = 20;
  }
  while(tabVoitures->nbVoituresFini <calculNbFini){
    msleep(500);
    afficheTab(tabVoitures, choix);
  }
  if (choix > 4) {
    triQualifications(tabVoitures, choix);
  }
  docRecap(tabVoitures, choix); //Pour avoir un document récapitulatif du tableau final
}

//Lance le programme
int main(int argc, char *argv[]){
  struct MemoirePartagee *tabVoitures;

  int shmid = shmget(1009, sizeof(tabVoitures), IPC_CREAT | 0666);
  tabVoitures = shmat(shmid, NULL, 0);
  int choix = 0;

  printf("Quelle est la taille du circuit ? (en km (int))");
  int tailleCircuit;
  scanf("%d", &tailleCircuit);

  tabVoitures->nbTourAFaire = calculNbTour(tailleCircuit);
 //printf("test: %d", tabVoitures->nbTourAFaire); //Nombre de tour à faire

  while(choix != 10){
    tabVoitures->nbVoituresFini = 0;

    printf("Que voulez-vous faire? \n");
```

```
printf("0 : pour lancer les essais 1 (P1)  \n");
printf("6 : pour lancer les essais 2 (P2)  \n");
printf("1 : pour lancer les essais 3 (P3)  \n");
printf("2 : pour lancer les qualifs 1 (Q1) \n");
printf("3 : pour lancer les qualifs 2 (Q2) \n");
printf("4 : pour lancer les qualifs 3 (Q3) \n");
printf("5 : pour lancer la course finale   \n");
printf("10 : pour quitter le programme \n");
  scanf("%d", &choix);

  switch(choix){
    case 0 : // P1 -> 1h30
    for(int i = 0; i < 20; i++){
      int pid = fork();
      if (pid < 0) {
        perror("ça forke pas\n");
        return -1;
      }
      else if (pid == 0) {
        srand(getpid());
        essaisEtQualifications(tabVoitures, i, 5400);
      }
    }
    fonctionPere(tabVoitures, 0); // C'est le père qui affiche
    break;
    case 6 : // P2 -> 1h30
    for(int i = 0; i < 20; i++){
      int pid = fork();
      if (pid < 0) {
        perror("ça forke pas\n");
        return -1;
      }
      else if (pid == 0) {
        srand(getpid());
        essaisEtQualifications(tabVoitures, i, 5400);
      }
    }
    fonctionPere(tabVoitures, 3);
    break;
    case 1 : //P3 -> 1h
    for(int i = 0; i < 20; i++){
      int pid = fork();
      if (pid < 0) {
        perror("ça forke pas\n");
        return -1;
      }
      else if (pid == 0) {
        srand(getpid());
        essaisEtQualifications(tabVoitures, i, 3600);
      }
    }
    fonctionPere(tabVoitures, 1);
    break;
    case 2 :
    for(int i = 0; i < 20; i++){
      int pid = fork();
      if (pid < 0) {
        perror("ça forke pas\n");
        return -1;
      }
      else if (pid == 0) {
        srand(getpid());
        essaisEtQualifications(tabVoitures, i, 1080);
      }
    }
    fonctionPere(tabVoitures, 20);
    break;
    case 3 :
    for(int i = 0; i < 15; i++){
      int pid = fork();
      if (pid < 0) {
        perror("ça forke pas\n");
        return -1;
      }
      else if (pid == 0) {
        srand(getpid());
        essaisEtQualifications(tabVoitures, i, 900);
      }
    }
```

```
    fonctionPere(tabVoitures, 15);
    break;
    case 4 :
    for(int i = 0; i < 10; i++){
     int pid = fork();
     if (pid < 0) {
      perror("ça forke pas\n");
      return -1;
     }
     else if (pid == 0) {
      srand(getpid());
      essaisEtQualifications(tabVoitures, i, 720);
     }
    }
    fonctionPere(tabVoitures, 10);
    break;
    case 5 :
    for(int i = 0; i < 20; i++){
     int pid = fork();
     if (pid < 0) {
      perror("ça forke pas\n");
      return -1;
     }
     else if (pid == 0) {
      srand(getpid());
      courseFinale(tabVoitures, i);
     }
    }
    fonctionPere(tabVoitures, 2);
    break;
   }
  }
  return 0;
}
```