```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

/* declare global variables including a table structure to hold scheduling information */
/* optional: define a function that finds the maximum of two integers */

/*****************************************************************/
void "PROCEDURE TO PRINT THE CONTENTS OF THE SCHEDULING TABLE"() {
 /* declare local variables */
 /* print table header */
 /* for each process */
  /* print the contents (id, arrival time, total_cycles) of each field of the table's index */
  /* if process has been scheduled ("done" field is 1, print other contents (start time, end time, turnaround time) */
 return;
}


/*****************************************************************/
void "PROCEDURE FOR OPTION #1"() {
 /* declare local variables */
 /* prompt for total number of processes */
 /* allocate memory for table to hold process parameters */
 /* for each process */
  /* prompt for process id, arrival time, and total cycle time */
 /* print contents of table */
 return;
}


/*****************************************************************/
void "PROCEDURE FOR OPTION #2"() {
 /* declare (and initilize when appropriate) local variables */
 /* for each process, reset "done" field to 0 */
 /* while there are still processes to schedule */
  /* initilize the earliest arrival time to INT_MAX (largest integer value) */
  /* for each process not yet scheduled */
   /* check if process has earlier arrival time than current earliest and update */
  /* set start time, end time, turnaround time, done fields for unscheduled process with earliest arrival time */
  /* update current cycle time and increment number of processes scheduled */
 /* print contents of table */
 return;
}


/*****************************************************************/
void "PROCEDURE FOR OPTION #3"() {
 /* declare (and initilize when appropriate) local variables */
 /* for each process, reset "done" field to 0 */
 /* while there are still processes to schedule */
  /* initilize the lowest total cycle time to INT_MAX (largest integer value) */
  /* for each process not yet scheduled */
   /* check if process has lower total cycle time than current lowest and has arrival time less than current cycle time a
nd update */
  /* set start time, end time, turnaround time, done fields for unscheduled process with lowest (and available) total cy
```

```
cle time */
  /* update current cycle time and increment number of processes scheduled */
 /* print contents of table */
 return;
}


/*****************************************************************/
void "PROCEDURE FOR OPTION #4"() {
 /* declare (and initilize when appropriate) local variables */
 /* for each process, reset "done", "total_remaining" and "already_started" fields to 0 */
 /* while there are still processes to schedule */
  /* initilize the lowest total remaining time to INT_MAX (largest integer value) */
  /* for each process not yet scheduled */
   /* check if process has lower total remaining time than current lowest and has arrival time less than current cycle ti
me and update */
  /* check if process already partially-scheduled */
   /* if so, set "start time", "already_started" fields of process with lowest (and available) total remaining cycle time *
/
  /* set end time, turnaround time of process with lowest (and available) total remaining cycle time */
  /* decrement total remaining time of process with lowest (and available) total remaining cycle time */
  /* if remaining time is 0, set done field to 1, increment cycle time and number of scheduled processes*/
 /* print contents of table */
 return;
}


/*****************************************************************/
void "PROCEDURE FOR OPTION #5"() {
 /* free the schedule table if not NULL */
 return;
}


/*****************************************************************/
int main() {
 /* declare local vars */
 /* while user has not chosen to quit */
  /* print menu of options */
  /* prompt for menu selection */
  /* call appropriate procedure based on choice--use switch statement or series of if, else if, else statements */
 } /* while loop */
  return 1; /* indicates success */
} /* end of procedure */
```