MAIN CODE

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

#define nPartitions 4
//total memory
#define memSize 1024
//one partition
#define partitionSize 256
#define true 1
#define false 0
#define nInterrupts 5

//state definitions
#define newState 10
#define ready 11
#define runMe 12
#define wait 13
#define terminate 14


#define scheduler 0
#define iowStart 1
#define iowComplete 2
#define jPoolLoad 3
#define jobHalt 4


// Here is the time quantum. Setting this to a large
// number will give the same results as FCFS.
#define TQ 5

struct pcb {
   int pid;
   int arrival;
   int first;
   int complete;
   int bursts;
   int state;
   int partitionNum;
   int jPC;
   char instructions[128];
};

// Computer Components
char userMem[memSize];
char partAvail[nPartitions];
int PC;
char IR;
char iFlags[nInterrupts];
```

```
// OS support variables.
struct pcb jobPool[16];
int pid = 0;
int currentJob = -1;
int TQinterrupt;
// This variable turns on Time Quantum interrupts
// for use with the round robin scheduler.
// mode allows for single variable schedule switching
// mode 0 = fcfs, 1 = sjf, 2 = rr
int mode = 2;

// Functions
void load(char *job, int address, int len);
void cpuMe2(int nJobs, int time);
int submitJobs(char *jFile);
int doTerminateQ(int nJobs);
void interruptService(int nJobs, int time);
int getNewPid(void);
void showMemory(void);
int firstComefirstServe(int nJobs, int time);
int shortestJobFirst(int nJobs, int time);
int roundRobinMe(int nJobs, int time);
void getRunStats(int nJobs);

int main()
{
   int j;

   int nJobs;
   char running = true;
   int failSafe = 0;
   int numComplete = 0;

   int time = 0;

   // Set the time quantum interrupt time.
   TQinterrupt = time;

   printf("Hello TV Land!\n");

   // Boot Sequence.
   // Clear memory.
   for(j=0;j<memSize;j++) {
      userMem[j] = '_';
   }
   // Clear interrupt flags.
   for(j=0;j<nInterrupts;j++) {
      iFlags[j] = false;
   }
   // Set partition available flags.
   for(j=0;j<nPartitions;j++) {
      partAvail[j] = true;
```

```
    }

    // Load jobs into our jobpool/pcb structure.  This does not mean they are in
    // memory yet.  We will fish them out of the pool and put them into
    // memory as they arrive based on arrival time.
    nJobs = submitJobs("jobs.txt");
    printf("\nPreparing to process: nJobs = %d \n", nJobs);

    while(running) {
        // Set flags.
        // Jobs are loaded into the job pool on a constant interval.
        if ((time%1)== 0) {
            iFlags[jPoolLoad] = true;
        }

        if (mode==2) {
            if (time == TQinterrupt) {
                iFlags[scheduler] = true;
                TQinterrupt = time + TQ;
            }
        }

        // Service interrupts.
        interruptService(nJobs, time);

        // Call CPU for user job.
        if (currentJob != -1) {
            cpuMe2(nJobs, time);
        }

        numComplete = doTerminateQ(nJobs);
        if (numComplete == nJobs)
            running = false;

        failSafe++;
        if (failSafe == 200)
            running = false;

        time++;
    }
    showMemory();
    getRunStats(nJobs);

    return 0;
}



int doTerminateQ(int nJobs)
{
    int j;
    int completed = 0;
```

```
    for(j=0;j<nJobs;j++) {
        if (jobPool[j].state == terminate) {
            completed++;
        }
    }
    return completed;
}

int submitJobs(char *jFile)
{
    FILE *in;
    int nJobs;
    int j;
    int nBursts;

    int arrivalTime;
    char instructions[129];

    in = fopen(jFile, "r");
    if (in != NULL) {
        printf("\nsubmitJobs: Input file %s opened \n", jFile);
        fscanf(in, "%d", &nJobs);
        printf("submitJobs: number of jobs = %d \n\n", nJobs);

        for(j=0;j<nJobs;j++) {
            // Read arrival time and instructions from job file.
            fscanf(in, "%d", &arrivalTime);
            printf("submitJobs: arrivalTime = %d \n", arrivalTime);
            fscanf(in, "%s", instructions);
            printf("submitJobs: instructions = %s \n", instructions);

            // Load the arrival time and instructions into the submitted jobs array of pcbs.
            jobPool[j].arrival = arrivalTime;
            jobPool[j].first = -1;
            jobPool[j].complete = -1;
            jobPool[j].state = newState;

            // Load pcb with relevant data.
            jobPool[j].pid = getNewPid();
            strcpy(jobPool[j].instructions, instructions);
            nBursts = strlen(jobPool[j].instructions);
            jobPool[j].bursts = nBursts;
        }
        fclose(in);
    }
    else {
        printf("submitJobs: Error - Job input file %s not opened \n", jFile);
    }

    return nJobs;
}
```

```
void load(char *job, int address, int len)
{
   int j;
   printf("job instructions = %s \n", job);
   for(j=0;j<len;j++) {
      if ((address + j) < 1024) {
         userMem[address + j] = job[j];
      }
      else {
         printf("Loader: Error - address (%d) exceeds memory size \n", address+j);
      }
   }
}

void cpuMe2(int nJobs, int time)
{
   // Fetch
   IR = userMem[PC];

   // Decode and Execute
   if (IR == 'a') {
      // assignment statement
   }
   else if (IR == 'i') {
      // input statement
      iFlags[iowStart] = true;
   }
   else if (IR == 'o') {
      // output statement
      iFlags[iowStart] = true;
   }
   else if (IR == 'w') {
      // wait statement
      iFlags[iowStart] = true;
   }
   else if (IR == 'e') {
      // end of program statement
      iFlags[jobHalt] = true;
   }

   // Increment program counter.
   PC++;
}

void interruptService(int nJobs, int time)
{
   int j, k;
   int address, len;
   int pNumber;
   int newJob;
   char found, looking;
```

```
    // Input, output, wait start flag
    if (iFlags[iowStart] == true) {
        // For now, reset the flag.
        iFlags[iowStart] = false;
    }
    // Input, output, wait complete flag
    if (iFlags[iowComplete] == true) {
        // For now, reset the flag.
        iFlags[iowComplete] = false;
    }
    // Job pool load.  Handles the "new state"
    if (iFlags[jPoolLoad] == true) {
        for(j=0;j<nJobs;j++) {
            // if a job has arrived do new state operations
            if ((time >= jobPool[j].arrival)&&(jobPool[j].state == newState)) {
                printf("\nservice jobpool load: time = %d  job arrival time = %d \n", time,
jobPool[j].arrival);

                // Load job into empty partition.
                // Find first empty partition.
                k = 0;
                found = false;
                looking = true;
                while(looking == true) {
                    if (partAvail[k] == true) {
                        found = true;
                        looking = false;
                    }
                    else {
                        k++;
                    }
                    if (k == nPartitions) {
                        looking = false;
                    }
                } // end while
                if (found == true) {
                    partAvail[k] = false;
                    // Load the code into the partition.
                    address = k * partitionSize;
                    len = jobPool[j].bursts;
                    load(jobPool[j].instructions, address, len);
                    // update the partition number.
                    jobPool[j].partitionNum = k;
                    // update the state of the process.
                    jobPool[j].state = ready;
                    // update the jobs pc to the program entry point.
                    jobPool[j].jPC = address;
                    // Set scheduler flag.
                    if (currentJob == -1) {
                        iFlags[scheduler] = true;
                    }
```

```
                printf("service jobpool load: Job in partition %d \n", k);
            }
            else {
                printf("service jobpool load: No partition available. \n");
            }
        }
    } // end for.

    // Reset job pool load interrupt flag.
    iFlags[jPoolLoad] = false;
  } // End if job pool interrupt flag was set.

  if (iFlags[jobHalt] == true) {
    printf("\nservice job complete: current job = %d    time = %d \n", currentJob, time);
    // Set current job to complete.
    jobPool[currentJob].state = terminate;
    // Set completion time.
    jobPool[currentJob].complete = time;
    // Release the partition for the job.
    pNumber = jobPool[currentJob].partitionNum;
    printf("PC = %d partitionSize = %d Freeing partition number %d \n", PC, partitionSize,
pNumber);
    partAvail[pNumber] = true;
    // Reset the job complete interrupt flag.
    iFlags[jobHalt] = false;
    // Set the scheduler flag.
    iFlags[scheduler] = true;

    if (mode==2) {
      TQinterrupt = time + TQ;
    }
  }

  // Scheduler flag
  if (iFlags[scheduler] == true) {
    printf("\nservice schedule. \n");
    // Set current job based on scheduling algorithm.
    if(mode == 0){
      newJob = firstComefirstServe(nJobs, time);
    }
    else if(mode == 1){
      newJob = shortestJobFirst(nJobs, time);
    }
    else if(mode == 2){
      newJob = roundRobinMe(nJobs, time);
    }


    printf("service schedule: newJob = %d \n", newJob);

    // If a new job has been scheduled.
    if (newJob != -1) {
```

```
            // Based on current job, execute context switch.
            if (currentJob == -1) {
               currentJob = newJob;
               jobPool[currentJob].state = runMe;
               // Set first time in run q.
               if (jobPool[currentJob].first == -1)
                  jobPool[currentJob].first = time;
            }
            else {
               // Save Context data.
               if (jobPool[currentJob].state != terminate) {
                  jobPool[currentJob].state = ready;
                  jobPool[currentJob].jPC = PC;
               }

               // Switch the context to the new job.
               currentJob = newJob;
               jobPool[currentJob].state = runMe;
               // Set first time in run q.
               if (jobPool[currentJob].first == -1)
                  jobPool[currentJob].first = time;
               PC = jobPool[currentJob].jPC;
            }
         }

         // Reset scheduler flag.
         iFlags[scheduler] = false;
      }
   }

   int firstComefirstServe(int nJobs, int time){
      int jArrivalTime = 10000;
      int myJob = -1;

      for (int i = 0; i < nJobs; i++) {
         if(jobPool[i].state == ready){
            if(jobPool[i].arrival < jArrivalTime){
               jArrivalTime = jobPool[i].arrival;
               myJob = i;
            }
         }
      }

      if(myJob != -1){
         printf("FCFS scheduled: myJob = %d \n", myJob);
      }

      return myJob;
   }

   int shortestJobFirst(int nJobs, int time){
      int jLength = 10000;
```

```
    int myJob = -1;

    for (int i = 0; i < nJobs; i++) {
        if(jobPool[i].state == ready){
            if(jobPool[i].bursts < jLength){
                jLength = jobPool[i].bursts;
                myJob = i;
            }
        }
    }

    if(myJob != -1){
        printf("SJF scheduled: myJob = %d \n", myJob);
    }
    return myJob;
}


int roundRobinMe(int nJobs, int time){
    int jProgCount = 10000;
    int myJob = -1;

    for (int i = 0; i < nJobs; i++) {
        if(jobPool[i].state == ready){
            printf("id= %d pc= %d realPC= %d \n", i, jobPool[i].jPC, (jobPool[i].jPC-
(jobPool[i].partitionNum*partitionSize)));
            if((jobPool[i].jPC-(jobPool[i].partitionNum*partitionSize)) < jProgCount){
                jProgCount = (jobPool[i].jPC-(jobPool[i].partitionNum*partitionSize));
                myJob = i;
            }
        }
    }

    if(myJob != -1){
        printf("RR scheduled: myJob = %d \n", myJob);
    }
    return myJob;
}

int getNewPid(void)
{
    int myPid;

    myPid = pid;
    pid++;
    return myPid;
}

void showMemory(void)
{
    int j;
```

```
    printf("\n\nUser Memory: \n");
    for(j=1;j<=memSize;j++) {
        printf("%c", userMem[j-1]);
        if ((j%64) == 0){
            printf("\n");
        }
    }
    printf("\n");
}


void getRunStats(int nJobs)
{
    int j;
    float sumFirst, sumComplete;
    float art, att;

    printf("\nProcessing Statistics: \n");
    printf("pid  bursts  arrival  first  complete \n");
    sumFirst = sumComplete = 0.0f;
    for(j=0;j<nJobs;j++) {
        printf("%d  %d  %d  %d  %d \n", jobPool[j].pid, jobPool[j].bursts, jobPool[j].arrival,
jobPool[j].first, jobPool[j].complete);
        sumFirst = sumFirst + (jobPool[j].first - jobPool[j].arrival);
        sumComplete = sumComplete + (jobPool[j].complete - jobPool[j].arrival);
    }
    art = sumFirst/nJobs;
    att = sumComplete/nJobs;
    printf("art: %.02f att: %.02f \n", art, att);
}
```

FCFS

Hello TV Land!

submitJobs: Input file jobs.txt opened
submitJobs: number of jobs = 5

submitJobs: arrivalTime = 100
submitJobs: instructions = aaiaaaaaaaaiaaaiaaie
submitJobs: arrivalTime = 105
submitJobs: instructions = aaaaaaaaae
submitJobs: arrivalTime = 106
submitJobs: instructions = aaaaaaaaaaaaaaae
submitJobs: arrivalTime = 109
submitJobs: instructions = aaiaaaaaaaaiaaae
submitJobs: arrivalTime = 110
submitJobs: instructions = aaaiaaaaae

Preparing to process: nJobs = 5

service jobpool load: time = 100 job arrival time = 100
job instructions = aaiaaaaaaaaiaaaiaaie
service jobpool load: Job in partition 0

service schedule.
FCFS scheduled: myJob = 0
service schedule: newJob = 0

service jobpool load: time = 105 job arrival time = 105
job instructions = aaaaaaaaae
service jobpool load: Job in partition 1

service jobpool load: time = 106 job arrival time = 106
job instructions = aaaaaaaaaaaaaae
service jobpool load: Job in partition 2

service jobpool load: time = 109 job arrival time = 109
job instructions = aaiaaaaaaaaiaaae
service jobpool load: Job in partition 3

service jobpool load: time = 110 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 111 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 112 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 113 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 114 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 115 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 116 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 117 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 118 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 119 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 120 job arrival time = 110
service jobpool load: No partition available.

service job complete: current job = 0   time = 120
PC = 20 partitionSize = 256 Freeing partition number 0

service schedule.
FCFS scheduled: myJob = 1
service schedule: newJob = 1

service jobpool load: time = 121 job arrival time = 110
job instructions = aaaiaaaaae
service jobpool load: Job in partition 0

service job complete: current job = 1   time = 130
PC = 266 partitionSize = 256 Freeing partition number 1

service schedule.
FCFS scheduled: myJob = 2
service schedule: newJob = 2

service job complete: current job = 2   time = 146
PC = 528 partitionSize = 256 Freeing partition number 2

service schedule.
FCFS scheduled: myJob = 3
service schedule: newJob = 3

service job complete: current job = 3   time = 162
PC = 784 partitionSize = 256 Freeing partition number 3

service schedule.
FCFS scheduled: myJob = 4

service schedule: newJob = 4

service job complete: current job = 4 time = 172
PC = 10 partitionSize = 256 Freeing partition number 0

service schedule.
service schedule: newJob = -1

User Memory:
aaaiaaaaaeaiaaaiaaie_____
_____
_____
_____
aaaaaaaaae_____
_____
_____
_____
aaaaaaaaaaaaaaae_____
_____
_____
_____
aaiaaaaaaaaiaaae_____
_____
_____
_____

Processing Statistics:
pid bursts arrival first complete
0 20 100 100 120
1 10 105 120 130
2 16 106 130 146
3 16 109 146 162
4 10 110 162 172
art: 25.60 att: 40.00
[1] + Done "/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-1hv3d3b4.vex" 1>"/tmp/Microsoft-MIEngine-Out-guamlcss.bcq"
lubuntu@lubuntu:~$

SJF

Hello TV Land!

submitJobs: Input file jobs.txt opened
submitJobs: number of jobs = 5

submitJobs: arrivalTime = 100
submitJobs: instructions = aaiaaaaaaaaiaaaiaaie
submitJobs: arrivalTime = 105
submitJobs: instructions = aaaaaaaaae
submitJobs: arrivalTime = 106
submitJobs: instructions = aaaaaaaaaaaaaaae
submitJobs: arrivalTime = 109
submitJobs: instructions = aaiaaaaaaaaiaaae
submitJobs: arrivalTime = 110
submitJobs: instructions = aaaiaaaaae

Preparing to process: nJobs = 5

service jobpool load: time = 100 job arrival time = 100
job instructions = aaiaaaaaaaaiaaaiaaie
service jobpool load: Job in partition 0

service schedule.
SJF scheduled: myJob = 0
service schedule: newJob = 0

service jobpool load: time = 105 job arrival time = 105
job instructions = aaaaaaaaae
service jobpool load: Job in partition 1

service jobpool load: time = 106 job arrival time = 106
job instructions = aaaaaaaaaaaaaaae
service jobpool load: Job in partition 2

service jobpool load: time = 109 job arrival time = 109
job instructions = aaiaaaaaaaaiaaae
service jobpool load: Job in partition 3

service jobpool load: time = 110 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 111 job arrival time = 110

service jobpool load: No partition available.

service jobpool load: time = 112 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 113 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 114 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 115 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 116 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 117 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 118 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 119 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 120 job arrival time = 110
service jobpool load: No partition available.

service job complete: current job = 0   time = 120
PC = 20 partitionSize = 256 Freeing partition number 0

service schedule.
SJF scheduled: myJob = 1
service schedule: newJob = 1

service jobpool load: time = 121 job arrival time = 110
job instructions = aaaiaaaaae
service jobpool load: Job in partition 0

service job complete: current job = 1   time = 130
PC = 266 partitionSize = 256 Freeing partition number 1

service schedule.
SJF scheduled: myJob = 4
service schedule: newJob = 4

service job complete: current job = 4   time = 140
PC = 10 partitionSize = 256 Freeing partition number 0

service schedule.
SJF scheduled: myJob = 2

service schedule: newJob = 2

service job complete: current job = 2 time = 156
PC = 528 partitionSize = 256 Freeing partition number 2

service schedule.
SJF scheduled: myJob = 3
service schedule: newJob = 3

service job complete: current job = 3 time = 172
PC = 784 partitionSize = 256 Freeing partition number 3

service schedule.
service schedule: newJob = -1

User Memory:
aaaiaaaaaeaiaaaiaaie_____
_____
_____
_____
aaaaaaaaae_____
_____
_____
_____
aaaaaaaaaaaaaaae_____
_____
_____
_____
aaiaaaaaaaaiaaae_____
_____
_____
_____

Processing Statistics:

pid	bursts	arrival	first	complete
0	20	100	100	120
1	10	105	120	130
2	16	106	140	156
3	16	109	156	172
4	10	110	130	140

art: 23.20 att: 37.60
[1] + Done "/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-rl5bfnig.ipy" 1>"/tmp/Microsoft-MIEngine-Out-s05azfhx.j11"
lubuntu@lubuntu:~$

RR

Hello TV Land!

submitJobs: Input file jobs.txt opened
submitJobs: number of jobs = 5

submitJobs: arrivalTime = 100
submitJobs: instructions = aaiaaaaaaaaiaaaiaaie
submitJobs: arrivalTime = 105
submitJobs: instructions = aaaaaaaaae
submitJobs: arrivalTime = 106
submitJobs: instructions = aaaaaaaaaaaaaaae
submitJobs: arrivalTime = 109
submitJobs: instructions = aaiaaaaaaaaiaaae
submitJobs: arrivalTime = 110
submitJobs: instructions = aaaiaaaaae

Preparing to process: nJobs = 5

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service schedule.
service schedule: newJob = -1

service jobpool load: time = 100 job arrival time = 100
job instructions = aaiaaaaaaaaiaaaiaaie
service jobpool load: Job in partition 0

service schedule.
id= 0 pc= 0 realPC= 0
RR scheduled: myJob = 0
service schedule: newJob = 0

service jobpool load: time = 105 job arrival time = 105
job instructions = aaaaaaaaae
service jobpool load: Job in partition 1

service schedule.
id= 1 pc= 256 realPC= 0
RR scheduled: myJob = 1
service schedule: newJob = 1

service jobpool load: time = 106 job arrival time = 106
job instructions = aaaaaaaaaaaaaae
service jobpool load: Job in partition 2

service jobpool load: time = 109 job arrival time = 109
job instructions = aaiaaaaaaaaiaaae
service jobpool load: Job in partition 3

service jobpool load: time = 110 job arrival time = 110
service jobpool load: No partition available.

service schedule.
id= 0 pc= 5 realPC= 5
id= 2 pc= 512 realPC= 0
id= 3 pc= 768 realPC= 0
RR scheduled: myJob = 2
service schedule: newJob = 2

service jobpool load: time = 111 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 112 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 113 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 114 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 115 job arrival time = 110
service jobpool load: No partition available.

service schedule.
id= 0 pc= 5 realPC= 5
id= 1 pc= 261 realPC= 5
id= 3 pc= 768 realPC= 0
RR scheduled: myJob = 3
service schedule: newJob = 3

service jobpool load: time = 116 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 117 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 118 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 119 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 120 job arrival time = 110
service jobpool load: No partition available.

service schedule.

id= 0 pc= 5 realPC= 5
id= 1 pc= 261 realPC= 5
id= 2 pc= 517 realPC= 5
RR scheduled: myJob = 0
service schedule: newJob = 0

service jobpool load: time = 121 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 122 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 123 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 124 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 125 job arrival time = 110
service jobpool load: No partition available.

service schedule.
id= 1 pc= 261 realPC= 5
id= 2 pc= 517 realPC= 5
id= 3 pc= 773 realPC= 5
RR scheduled: myJob = 1
service schedule: newJob = 1

service jobpool load: time = 126 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 127 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 128 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 129 job arrival time = 110
service jobpool load: No partition available.

service jobpool load: time = 130 job arrival time = 110
service jobpool load: No partition available.

service job complete: current job = 1   time = 130
PC = 266 partitionSize = 256 Freeing partition number 1

service schedule.
id= 0 pc= 10 realPC= 10
id= 2 pc= 517 realPC= 5
id= 3 pc= 773 realPC= 5
RR scheduled: myJob = 2
service schedule: newJob = 2

service jobpool load: time = 131 job arrival time = 110
job instructions = aaaiaaaaae
service jobpool load: Job in partition 1

service schedule.
id= 0 pc= 10 realPC= 10
id= 3 pc= 773 realPC= 5
id= 4 pc= 256 realPC= 0
RR scheduled: myJob = 4
service schedule: newJob = 4

service schedule.
id= 0 pc= 10 realPC= 10
id= 2 pc= 522 realPC= 10
id= 3 pc= 773 realPC= 5
RR scheduled: myJob = 3
service schedule: newJob = 3

service schedule.
id= 0 pc= 10 realPC= 10
id= 2 pc= 522 realPC= 10
id= 4 pc= 261 realPC= 5
RR scheduled: myJob = 4
service schedule: newJob = 4

service job complete: current job = 4   time = 150
PC = 266 partitionSize = 256 Freeing partition number 1

service schedule.
id= 0 pc= 10 realPC= 10
id= 2 pc= 522 realPC= 10
id= 3 pc= 778 realPC= 10
RR scheduled: myJob = 0
service schedule: newJob = 0

service schedule.
id= 2 pc= 522 realPC= 10
id= 3 pc= 778 realPC= 10
RR scheduled: myJob = 2
service schedule: newJob = 2

service schedule.
id= 0 pc= 15 realPC= 15
id= 3 pc= 778 realPC= 10
RR scheduled: myJob = 3
service schedule: newJob = 3

service schedule.
id= 0 pc= 15 realPC= 15
id= 2 pc= 527 realPC= 15
RR scheduled: myJob = 0

service schedule: newJob = 0

service job complete: current job = 0 time = 170
PC = 20 partitionSize = 256 Freeing partition number 0

service schedule.
id= 2 pc= 527 realPC= 15
id= 3 pc= 783 realPC= 15
RR scheduled: myJob = 2
service schedule: newJob = 2

service job complete: current job = 2 time = 171
PC = 528 partitionSize = 256 Freeing partition number 2

service schedule.
id= 3 pc= 783 realPC= 15
RR scheduled: myJob = 3
service schedule: newJob = 3

service job complete: current job = 3 time = 172
PC = 784 partitionSize = 256 Freeing partition number 3

service schedule.
service schedule: newJob = -1

User Memory:
aaiaaaaaaaaiaaaiaaie_____
_____
_____
_____
aaaiaaaaae_____
_____
_____
_____
aaaaaaaaaaaaaae_____
_____
_____
_____
aaiaaaaaaaaiaaae_____
_____
_____
_____

Processing Statistics:

pid	bursts	arrival	first	complete
0	20	100	100	170
1	10	105	105	130
2	16	106	110	171
3	16	109	115	172
4	10	110	135	150

art: 7.00 att: 52.60
[1] + Done "/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-uhkipmu1.c1h" 1>"/tmp/Microsoft-MIEngine-Out-4u3ajkam.5fi"