```
int ksleep(int event)
{
  int sr = int_off();
    printf("proc %d going to sleep on event=%x\n", running->pid, event);

    running->event = event
    running->status = SLEEP;

    enqueue(&sleepList, running);//denque the current proc into the sleep queue
    tswitch();//call switch which deques new process

  int_on(sr);
}

int kwakeup(int event)
{
  int sr = int_off();
  PROC *temp, *proc;

  temp = 0;

  while( proc =dequeue(&sleepList))//while there is a proc to dequeue from the sleep
queue
  {
    if(proc->event == event)//if we have found the correct proc to wakeup
    {
      printf("wakeup proc #%d on event #%d\n", proc->pid, event);
      proc->status = READY;
      enqueue(&readyQueue, proc);
    }

    else
    {
      enqueue(&temp, proc);//if we have not found any procs that match
    }

  }

  sleepList = temp;

  int_on(sr);
}


int kexit(int exitCode)
{
  int i, wakeup;
  PROC *p;

  wakeup = sendChild();  // give children to P1

  running->exitCode = exitCode;
  running->status = ZOMBIE;//set it as zombie

  kwakeup((int)running->parent);//wkeup the zomie chid's parent proc
  tswitch();
}

int kwait(int *status)
{
  int i, found = 0;
  PROC *p;

  if(!running->child)//nothing to wait
```

```
  {
    printf("Has no child.\n");
    return -1;
  }

  while(1){
    // if can find a ZOMBIE child
    {
      for(i=1;i<NPROC;i++)//looks through current procs and find the children of the
proc
      {
        p=&proc[i];
        if(p->status != FREE && p->ppid == running->pid)
        {
          found = 1;
          if(p->status == ZOMBIE)//child has died
          {
            *status = p->exitCode;
            p->status = FREE;
            enqueue(&freeList, p);//sends the deceased child to the free list
            return(p->pid);
          }
        }
      }
    }

    ksleep((int)running);
  }

}
int sendChild()
{
  PROC *p = 0;
  int wakeupP1=0;
  /* send children (dead or alive) to P1's orphanage */
  for (int i = 1; i < NPROC; i++)
  {
    p = &proc[i]; //set temp to proc[i] to see if a child of running proc
    if (p->status != FREE && p->ppid == running->pid)
    {
      p->ppid = 1;
      p->parent = &proc[1];
      wakeupP1++;
    }
  }

  return wakeupP1;
}
```