Princeton University
COS 217: Introduction to Programming Systems
Fall 2003 Final Exam Answers

The exam was a three-hour, open-book, open-notes exam.

Question 1

```
   .section ".text"
   .align 4
   .global clearkeys
clearkeys:
   save %sp, -96 %sp
   mov  %i0, %l0
loop:
   cmp  %l0, 0
   be   endloop
   nop
   st   0, [%l0]
   ld   [%l0 + 4],  %o0
   call clearkeys
   nop
   ld   [%l0 + 8], %l0
   ba   loop
   not
endloop:
   ret
   restore
```

Question 2

```
int simulate(int n, unsigned int code[])
{
   union
   {
      unsigned int uiInstr;

      struct Instr3a
      {
         unsigned int uiOp: 2;
         unsigned int uiRd: 5;
         unsigned int uiOp3: 6;
         unsigned int uiRs1: 5'
         unsigned int uiI: 1;
         unsigned int uiAsi: 8;
         unsigned int uiRs2: 5;
      } sInstr3a;

      struct Instr3b
      {
         unsigned int uiOp: 2;
         unsigned int uiRd: 5;
         unsigned int uiOp3: 6;
         unsigned int uiRs1: 5'
         unsigned int uiI: 1;
         unsigned int uiImm13: 13;
      } sInstr3b;
   } uInstr;
```

```
   unsigned int ui;
   unsigned int uiReturnValue;

   puiRegs = (unsigned int*)calloc(32, sizeof(unsigned int));
   assert(puiRegs != NULL);

   for (ui = 0; ui < n; ui++)
   {
      uInstr.uiInstr = code[ui];
      assert(uInstr.sInstr3a.uiOp == 2);
      assert(uInstr.sInstr3a.uiOp3 == 0);
      if (uInstr.sInstr3a.uiI == 0)
         puiRegs[uInstr.sInstr3a.uiRd] =
            puiRegs[uInstr.sInstr3a.uiRs1] + puiRegs[uInstr.sInstr3a.uiRs2];
      else
      {
         puiRegs[uInstr.sInstr3b.uiRd] =
            puiRegs[uInstr.sInstr3b.uiRs1] + suInstr.sInstr3b.uiImm13;
      }
   }

   uiReturnValue = puiRegs[1];
   free(puiRegs);
   return uiReturnValue;
}
```

Question 3: File search.c

```
#include <stdio.h>
#include “maze.h”

void search(Maze_T oMaze, Place_T oPlace)
{
   if (! Maze_placeIsVisited(oMaze, oPlace))
   {
      if (Maze_placeIsDestination(oMaze, oPlace))
      {
         Maze_print(oMaze);
         exit(0);
      }
      Maze_markPlaceAsInPath(oMaze, oPlace);

      /* Pass function pointer so Maze can callback. */
      Maze_searchAdjacencies(oMaze, oPlace, search);

      Maze_markPlaceAsNotInPath(oMaze, oPlace);
   }
}

int main(void)
{
   Maze_T oMaze;
   Place_T oPlace;

   Maze_read(oMaze);
   for (oPlace = Place_new(oMaze);
        Maze_placeIsValid(oMaze, oPlace);
        Maze_nextPlace(oMaze, oPlace))
   {
      if (Maze_placeIsStart(oMaze, oPlace))
          search(oMaze, oPlace);
   }
   printf(“No valid path\n”);
   Maze_free(oMaze);
   Place_free(oMaze, oPlace);
}
```

Question 3: File maze.h

```
#ifndef MAZE_INCLUDED
#define MAZE_INCLUDED

typedef struct Maze *Maze_T;
typedef struct Place *Place_T;

Maze_T Maze_new();
/* Return a new Maze_T. */

void Maze_free(Maze_T oMaze);
/* Free oMaze. */

void Maze_read(Maze_T oMaze);
/* Read a new maze into oMaze. */

void Maze_print(Maze_T oMaze);
/* Print oMaze. */

int Maze_ placeIsStart(Maze_T oMaze, Place_T oPlace);
/* Return 1 (TRUE) iff oPlace (a place within oMaze) is a start place. */

int Maze_ placeIsDestination(Maze_T oMaze, Place_T oPlace);
/* Return 1 (TRUE) iff oPlace (a place within oMaze) is a destination place. */

int Maze_ placeIsVisited (Maze_T oMaze, Place_T oPlace);
/* Return 1 (TRUE) iff oPlace (a place within oMaze) has been visited. */

int Maze_placeIsValid(Maze_T oMaze, Place_T oPlace);
/* Return 1 (TRUE) iff oPlace is a valid place within oMaze. */

void Maze_nextPlace(Maze_T oMaze, Place_T oPlace);
/* Change oPlace (a place within oMaze) so it denotes the next place in oMaze, or
   an invalid place if there is no next place. */

void Maze_markPlaceAsInPath(Maze_T oMaze, Place_T oPlace);
/* Mark oPlace (a place within oMaze) as visited. */

void Maze_markPlaceAsNotInPath(Maze_T oMaze, Place_T oPlace);
/* Mark oPlace (a place within oMaze) as not in the path. */

void Maze_searchAdjacencies(Maze_T oMaze, Place_T oPlace,
   void (*search)(Maze_T oMaze, Place_T oPlace));
/* Search all of the places within oMaze that are adjacent to oPlace. */

Place_T Place_new(Maze_T oMaze);
/* Return a new Place_T that denotes the first place in oMaze. */

void Place_free(Maze_T oMaze, Place_T oPlace);
/* Free oPlace, a place within oMaze. */

/* It is a checked runtime error for oMaze, oPlace, or search to be NULL in any
   function. */

#endif
```

Question 4

```
#include <unistd.h>
#include <sys/wait.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
   int iPid = fork();
   if (iPid == -1) { perror(argv[0]); exit(1); }
   if (iPid == 0)
   {
      char *ppcArgv[3];
      struct rlimit sRlimit;
      int iRet;

      sRlimit.rlim_cur = 600;
      sRlimit.rlim_max = 600;
      iRet = setrlimit(RLIMIT_CPU, &sRlimit);
      if (iRet == -1) { perror(argv[0]); exit(1); }

      ppcArgv[0] = "/u/mydir/slowplayer";
      ppcArgv[1] = argv[1];
      ppcArgv[2] = NULL;

      execvp(ppcArgv[0], ppcArgv);
      perror(argv[0]);
      exit(1);
   }
   wait(NULL);
   return 0;
}
```

Question 5: Part A

```
0

Explanation:  The program redirects stdout to a pipe.  So no data is written to file
output.
```

Question 5: Part B

```
1
2
3
5
8
```