Will Clingan
CIS 657, Midterm
Dr. Mo
2025 May 20

#Q1

```c
/* xsh_create.c - xsh_create */

#include <xinu.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
extern process runforever(void);

/*------------------------------------------------------------------------
 * xsh_create - Shell command to create a new process.  *
 *------------------------------------------------------------------------
 */

shellcmd xsh_create(int nargs, char *args[]) {
    int priority;
    pid32 pid;

    /* Print help menu and system utilization. */
    if (nargs == 2 && strcmp(args[1], "--help") == 0) {
        printf("Usage: create <priority>\n");
        printf("Creates a new process at the specified priority which loops forever.\n");
        printf("If priority is less than 20, a warning will be displayed.\n");
        printf("If priority is less than 10, the process may make the shell unresponsive.
\n");
        return 0;
    }

    if (nargs != 2) {
        fprintf(stderr, "Usage: create <priority>\n");
        return 1;
    }

    /* Validate user input. */
    priority = atoi(args[1]);
    if (priority <= 0) {
        fprintf(stderr, "Invalid priority: %s\n", args[1]);
        return 1;
    }

    /* Prompt user w/ WARNING respective of process' priority. */
    if (priority < 20 && priority >= 10) {
        printf("WARNING: Priority values lower than 20 are typically reserved for shell/system processes.\n");
        printf("                Creating a user process at this priority may interfere with shell responsiveness.\n");
    }
    if (priority < 10) {
        printf("WARNING: Priority values lower than 10 are reserved for critical system processes.\n");
        printf("                Creating a user process at this priority may make the shell or system unresponsive.\n");
    }

    /* Create a process that prints its PID and loops forever */
    pid = create(runforever, 1024, priority, "runforever", 0);
    if (pid == SYSERR) {
        fprintf(stderr, "Failed to create process.\n");
        return 1;
    }
    resume(pid);
    printf("Created process with PID %d at priority %d\n", pid, priority);

    return 0;
}
```

#Q2

```
/* xsh_createsleep.c - xsh_createsleep */

#include <xinu.h>
extern process runafterwait(void);

/*------------------------------------------------------------------------
 * xsh_createsleep - Shell command to sleep a process.  *
 *------------------------------------------------------------------------
 */

shellcmd xsh_createsleep(int nargs, char *args[]) {
    int prio = 20;
    if (nargs == 2) {
        prio = atoi(args[1]);
    }
    resume(create((void *)runafterwait, 1024, prio, "runafterwait", 0));
    return 0;
}
```

#Q3

```
/* xsh_psready.c - xsh_psready */

#include <xinu.h>

/*------------------------------------------------------------------------
 * xsh_psready - Shell command to print the PID Table.  *
 *------------------------------------------------------------------------
 */

shellcmd xsh_psready(int nargs, char *args[]) {
    int32 i;
    struct procent *prptr;

    kprintf("Ready Processes:\n");
    kprintf("%-10s %-10s %-10s\n", "PID", "State", "Priority");
    for (i = 0; i < NPROC; i++) {
        prptr = &proctab[i];
        if (prptr->prstate == PR_READY) {
            kprintf("%-10d %-10s %-10d\n", i, "READY", prptr->prprio);
        }
    }
    return 0;
}
```

#Q4

```
/* xsh_wait.c - xsh_wait */

#include <xinu.h>
extern sid32 globalsemaphore;
extern process waiter(void);

/*------------------------------------------------------------------------
 * xsh_wait - Shell command to put a process into the wait pattern.  *
 *------------------------------------------------------------------------
 */

shellcmd xsh_wait(int nargs, char *args[]) {
    resume(create((void *)waiter, 1024, 20, "waiter", 0));
    return 0;
}
```

#Q5:

```
/* xsh_signaln.c - xsh_signaln */
```

```c
#include <xinu.h>
extern sid32 globalsemaphore;

/*------------------------------------------------------------
 * xsh_signaln - Shell command to signal a semaphore. *
 *------------------------------------------------------------
 */

shellcmd xsh_signaln(int nargs, char *args[]) {
    int n = 1;
    if (nargs == 2) {
        n = atoi(args[1]);
    }
    signaln(globalsemaphore, n);
    kprintf("Signaled semaphore %d times\n", n);
    return 0;
}
```

#Q6

```c
/* xsh_resumen.c - xsh_resumen */

#include <xinu.h>

/*------------------------------------------------------------
 * xsh_resumen - Shell command to resume process. *
 *------------------------------------------------------------
 */

shellcmd xsh_resumen(int nargs, char *args[]) {
    int i;
    pid32 maxprio_pid = -1;
    int maxprio = -1;
    struct procent *prptr;

    if (nargs < 2) {
        kprintf("Usage: resumen <pid1> <pid2> ...\n");
        return 1;
    }

    for (i = 1; i < nargs; i++) {
        pid32 pid = atoi(args[i]);
        prptr = &proctab[pid];
        if (prptr->prprio > maxprio) {
            maxprio = prptr->prprio;
            maxprio_pid = pid;
        }
        resume(pid);
    }

    if (maxprio_pid != -1 && proctab[maxprio_pid].prstate == PR_READY) {
        resched();
    }
    return 0;
}
```