CS240 Final Exam Answers 08/04/2016

P1(a) 12 pts

```
int main(int argc, char **argv)
{
FILE *fp1, *fp2;
char c;

  fp1 = fopen(argv[1],"r");
  fp2 = fopen(argv[2],"w");

  while((c = fgetc(fp1)) != EOF)
        fputc(c, fp2);
}
```

Correct use of argv
4 pts

Correct use of loop of read/write
4 pts

Other parts of code
4 pts

P1(b) 12 pts

execlp() makes use of the environment variable PATH where the directories to be searched are listed
separated by colons. The environment variables can be obtained from the third argument passed to main().
5 pts

execlp() looks at the directories listed in PATH, checking one-by-one to determine if the
file to be executed exists. If so, it prepends the command (e.g., ls) with the directory (/bin) to
create the full pathname (/bin/ls). It then does what execl() does (call execve()).
5 pts

If the requested command is not found, execlp() prints an error message and returns.
2 pts

P1(c) 12 pts

int fprintf(FILE *fp, const char *str, ...);
In the above, const can be omitted without penalty.
4 pts

fprintf() parses the format string pointed to by str in a loop until the end of str is reached and looks
for the symbol %. For each such occurrence, it reads the characters following % (e.g., d, f, s, c) to
determine what/how to print in a switch statement (or using if-else). It calls arg_va with type matching
that of % to access the next argument. It formats and prints the argument to the file pointed to by fp
using the system call write().
4 pts

In the loop, if it encounters special symbols escaped by \ (e.g., \n) or literal symbols, they are output
accordingly using write().
2 pts

Based on the code structure outlined above, the likely outcome is that the sixth argument will be ignored
since only five calls to va_arg() will be made.
2 pts

P2(a) 12 pts

```
int signal(int x, void (* func)(int));
```
6 pts

Inside signal(): `(* func)(x);`
In the above func(x) is fine too.
6 pts

P2(b) 12 pts

const informs the C compiler that the argument should not be allowed to change. An attempt to change it
will result in error.
4 pts

Without the writer of the function execlp() specifying the argument with const, a programmer who calls
execlp() will not know if the code of execlp() will modify string pointed by file. Hence, to be sure, the
programmer would need to create a copy of the string whose address is then passed to execlp() so that
the original string is not corrupted by execlp().
5 pts

As indicated above, if corruption is an issue, make a copy of the string variable and pass a pointer to
the copy. To check for overflow, a canary may be inserted in the copy

but that's a bit of an overkill.
3 pts

P2(c) 12 pts

char **argv specifies that argv contains an address at whose location there is an address that then
points to memory containing char. char *argv[] states that the 1-D array argv[] contains pointers to char.
Since a 1-D array is char *, they are as declaration consistent with each other.
4 pts

An example layout assuming 32-bit arch (var: symbol, addr: address, val: content):

var	addr	val
	5010	NULL
	5009	t
	5008	a
	5007	d
	5006	.
	5005	t
	5004	u
	5003	p
	5002	t
	5001	u
	5000	o
	...	
	4009	NULL
	4008	t
	4007	a
	4006	d
	4005	.
	4004	t
	4003	u
	4002	p
	4001	n
	4000	i
	...	
	3002	NULL
	3001	p
	3000	c
	...	
	2008	5000
	2003	4000
	2000	3000
	...	
argv	1000	2000

Deduct if they fail to show what an array of strings is and how it is represented in memory.
8 pts

Deduct points if pointer arithmetic or other pertinent details are incorrect.
4 pts

P3 28 pts

```
main() {
char a[1000];
int b;

// empty list
xstart = NULL;
xend = NULL;

 while(1) {
  scanf("%s %d", a, &b);

  // empty list
  if (xstart == NULL) {
        xstart = (mystuff_t *) malloc(sizeof(mystuff_t));
        xstart->name = (char *) malloc(strlen(a));
        strcpy(xstart->name, a);
        xstart->value = b;
        xstart->next = NULL;
        xend = xstart;
  }
  else {
        xend->next = (mystuff_t *) malloc(sizeof(mystuff_t));
        (xend->next)->name = (char *) malloc(strlen(a));
        strcpy((xend->next)->name, a);
        (xend->next)->value = b;
        (xend->next)->next = NULL;
        xend = xend->next;
 }
}
```

The while-loop construct is not necessary. For example, just reading once from standard input is fine.

Deduct points if the initial case (empty list) is not properly considered.
7 pts

Deduct points if logic of linking lists and referencing structures has problems.
10 pts

Deduct points if malloc has problems.
7 pts

For other coding problems.
4 pts

Bonus 10 pts

Stack smashing occurs if the memory allocated for a[100] in the stack area (i.e., frame) of the
function is overwritten to overflow into the memory above which may corrupt the return address of the
function.
5 pts

gcc may insert canary code, meaning a random number is placed below the return address, so that if
overflow occurs the random number changes. When returning from the function call, gcc inserts code that
checks the canary (i.e., random value) against what it actually finds. If they are not equal, an error
is reported at run-time.
5 pts