CSED211 Homework #4, Due Dec. 4, 2019

1. Exercise 10.6 on page 950.

   On entry, descriptors 0-2 are already open. The open function always returns the lowest possible descriptor, so the first two calls to open return descriptors 3 and 4. The call to the close function frees up descriptor 4, so the final call to open returns descriptor 4, and thus the output of the program is "fd2 = 4".

2. Exercise 10.9 on page 950

   Before the call to execve(), the child process opens foo.txt as descriptor 3, redirects stdin to foo.txt, and then (here is the kicker) closes descriptor 3:

```
if (Fork() == 0) { /* child */
    fd = Open(''foo.txt'', O_RDONLY, 0); /* fd == 3 */
    Dup2 (fd, STDIN_FILENO);
    Close(fd);
    Execve(''fstatcheck'', argv, envp);
}
```

   When fstatcheck begins running in the child, there are exactly three open files, corresponding to descriptors 0, 1, and 2, with descriptor 1 redirected to foo.txt.

3. Exercise 9.11 on page 912

   A. 00 0010 0111 1100

VPN:	0x9
TLBI:	0x1
TLBT:	0x2
TLB hit?	N
page fault?	N
PPN:	0x17

   B. 0101 1111 1100

CO:	0x0
CI:	0xf
CT:	0x17
cache hit?	N
cache byte?	--

4. Exercise 9.13 on page 914

   A. 00 0000 0100 0000

   B. VPN: 0x1

TLBI: 0x1

TLBT: 0x0

TLB hit? N

page fault? Y

PPN: --

C. n/a

D. n/a

5. Exercise 9.14 on page 915

Sample code

```
1    #include "csapp.h"
2
3    /*
4    * mmapwrite - uses mmap to modify a disk file
5    */
6    void mmapwrite (int fd, int len)
7    {
8        char *bufp;
9
10       bufp = Mmap (NULL, len, PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0);
11       bufp[0] = 'J';
12   }
13
14   /* mmapwrite driver */
15   int main (int argc, char **argv)
16   {
17       int fd;
18       struct stat stat;
19
20       /* check for required command line argument */
21       if (argc != 2) {
22           printf ("usage: %s <filename>₩n", argv[0]);
23           exit (0);
24       }
25
26       /* open the input file and get its size */
27       fd = Open (argv[1], O_RDWR, 0);
28       fstat (fd, &stat);
29       mmapwrite (fd, stat.st_size);
30       exit (0);
31   }
```

6. Exercise 9.15 on page 915

Request	Block size (decimal bytes)	Block header (hex)
`malloc(3)`	8	`0x9`
`malloc(11)`	16	`0x11`
`malloc(20)`	24	`0x19`
`malloc(21)`	32	`0x21`

7. Exercise 9.19 on page 916

Here are the true statements. The observation about the equivalence of first fit and best fit when the list is ordered is interesting.

1. (a) In a buddy system, up to 50% of the space can be wasted due to internal fragmentation.

2. (d) Using the first-fit algorithm on a free list that is ordered according to increasing block sizes is equivalent to using the best-fit algorithm.

3. (b) Mark-and-sweep garbage collectors are called conservative if they treat everything that looks like a pointer as a pointer.