

(out of 75 points)

1. What is the difference between `_exit()` and `exit()` and `_Exit()`?

7/5/3/0

2. In APUE, Section 8.5, a process can terminate normally in five ways, and we discussed three of these in the lecture notes: Provide a code snippet example of these termination conditions.

a)

3/1/0

b)

3/1/0

c)

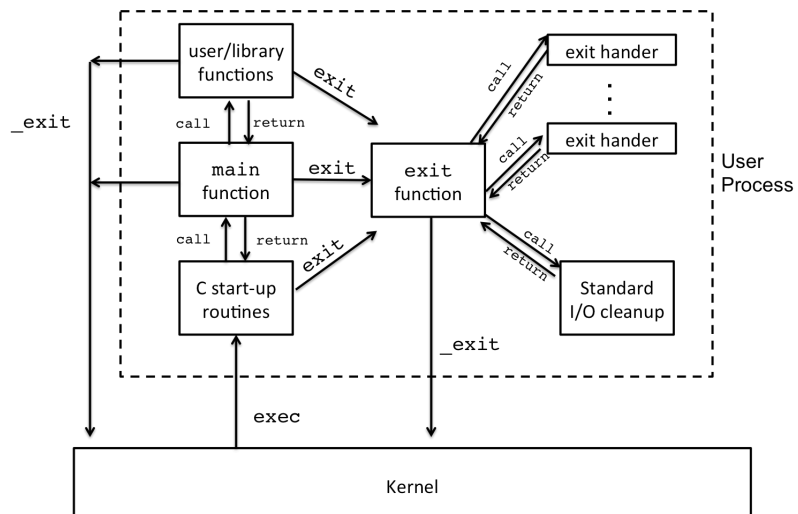
3/1/0

3. In the diagram below, place a circle along the exit path of the following program:

```

void fun(){
    _exit(1);
}

int main(){
    fun();
    exit(0);
}
    
```

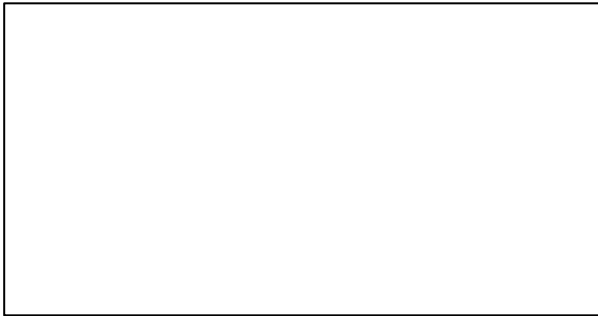


5/3/1/0

4. Consider the following programs, what are their output, and why? Be sure to discuss I/O buffering and exit condition.

a)

6/4/2/0

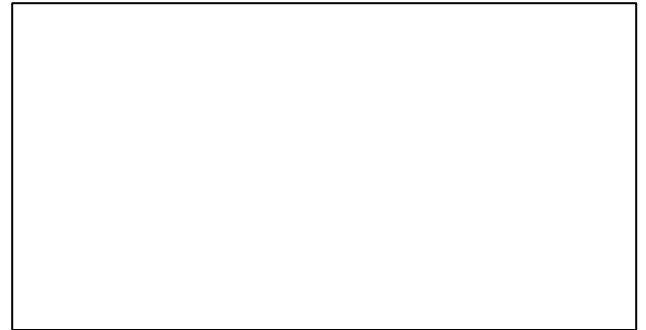


```
int main(){  
  
    fprintf(stdout, "Hello World!");  
  
    return 0;  
}
```

b)

6/4/2/0

```
int main(){  
  
    fprintf(stdout, "Hello World!");  
    exit(0);  
}
```



c)

6/4/2/0

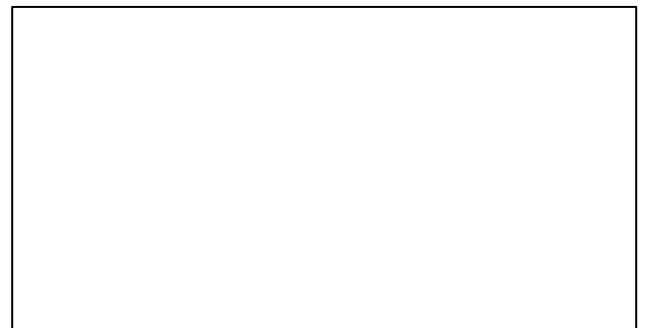


```
int main(){  
  
    fprintf(stdout, "Hello World!");  
    _Exit(0)  
}
```

d) `int main(){`

6/4/2/0

```
    fprintf(stderr, "Hello World!");  
    _exit(0)  
}
```



5. Match each of the buffer settings to their mode options to `setvbuf()`

5/3/1/0 `_IONBF` _____ a) unbuffered
 `_IOFBF` _____ b) line buffered
 `_IOLBF` _____ c) fully buffered

6. What is the difference between line buffered and fully buffered?

5/3/1/0

7. Why does the following code snippet properly check for a failed call to `execv()`

7/5/2/0

```
int main(){
    char * ls_args[2] = { "/bin/ls", NULL} ;

    execv( ls_args[0], ls_args);
    perror("execve failed");

    _exit(1); //failure
}
```

8/6/3/0 8. Consider setting up an `argv` array to exec the program:
`find . -type d -name ic221` . Fill in the `argv` declaration for these options:

`char * argv = {` `};`

`execv(argv[0], argv);`

9. The `argv` array must be NULL terminated, why? How does this relate to `argc`, the number of arguments value that is passed to `main()`?

5/3/0