I.1

1) // Empty // Opens fd1 descriptor with writeonly flag and offset 0

2) // Empty // Opens fd2 descriptor with writeonly flag and offset 0

3) AAAAAA // Write 6 'A' to file through fd1 and fd1 offset changes to 6

4) BBBAAA // Overwrites the file with 3 'B' through fd2 since fd2 offset is 0 and fd2 offset changes to 3 after the write

5) BBBAAA // dup2 functions closes previously opened fd2 file descriptor and duplicates fd1 file descriptor to fd2 file descriptor with same offset 6

6) BBBAAABBB // Appends 3 'B' to the end of file through fd2 since fd2 offset is same as fd1

7) BBBAAABBB // Both file descriptors are closed.

I.2

The Pipe '|' operator connects stdout of the first process to the stdin of the second process. Pipe actually creates two file descriptor , one referring to read end and another to write end. Pipe in bash opens the file descriptor in blocking mode. So until writing process terminates or the buffer gets full, it won't be read by the reading process. Due to this, there is delay in getting a output.

I.3

```
int count_sym_links(const char *dir)
{
        DIR *dir = opendir(dir);
        int count = 0;
        for (dirent dP = readdir(dir); dP; dP = readdir(dir))
=>          ^ 'struct' keyword is missing
        {
                const char *name = dP->d_name;
                struct stat *statP;
=>              ^ should allocate struct using malloc before using it
                char dir_name[strlen(dir) + 1 + strlen(name)];
=>                    ^ dir_name is declared but the value is NULL
                sprintf("%s/%s", dir, name);
=>              ^ 'sprintf' returns string which is not used
                stat(dir_name, statP);
=>              ^ should use lstat instead of stat to check a directory is symlink
                if (S_ISLNK(statP->st_mode))
```

```
                        count++;
        }
        return count;
}
```

I.4

| | Data1 R | Data1 W | Data2 R | Data2 W |
|---|---|---|---|---|
| u1 runs exec1 | Y | N | Y | Y |
| u2 runs exec1 | Y | Y | Y | N |
| u1 runs exec2 | Y | N | Y | Y |
| uw runs exec2 | Y | N | Y | Y |

I.5

1) cd  : Current directory is associated with a process. It can be run as external command but it does not change the  current working directory of the shell. So it must be run as builtin Command

2) pwd : This can be run as external command as the new process inherits the running shell's current working directory.

3) exec: This command replaces the current working shell by another program. So it must be run as builtin command as the parent process must be shell

4) exit: This can be run as external command as we can kill the parent process from the child process by signal.

I.6

1) Invalid : writing non-empty data to a file descriptor changes the file offset but it does not affect file pointer associated with the file descriptor
2) Invalid : The default permission mode for created file is readonly. So even the owner of the file cannot remove the file without changing the permission.
3) Invalid : The userid of the file is set to the effective userid of the process. But the groupid of the file  is set to the group id of the parent directory or the effective group id of the process.
4) Invalid : If the file is moved between same partition, there is no movement of data. But if file is moved between different partition, the data is moved from one partition to another.