Question 1

Working of the program

The program forks into a child and a parent process. The child process calls func() with the section ‘A’ to enable the program to compute only entries of section ‘A’ students. The parent process waits for the child to end the computations using waitpid(pid of child) and then calls the same function using ‘B’ as the argument.

System calls used

- **Open**
  - Syntax: int open (const char* Path, int flags [, int mode])
  - Path: path to the file
  - Flag like 0_RDONLY :read only
  - Return -1 if file opening is unsuccessful.

- **Read**
  - Syntax :size_t read (int fd, void* buf, size_t cnt);
  - Fd is the file descriptor of the file to be read.
  - Buf is the space to load cnt bytes of file.
  - It returns 0 when it reaches eof.
  - It returns -1 when there is an error in reading.
  - It returns no bytes when reading is successful.
- **Close**
  - Syntax : int close(int fd);
  - Fd is the file descriptor of file to close.
  - It returns 0 when the file is successfully closed.
  - It returns -1 when there is an error in closing a file.

- **Fork**
  - Syntax : fork()
  - It takes no arguments and returns an integer value.
    - **Negative** Value signifies unsuccessful creation of a child process.
    - **Zero** is the value returned to a child.
    - **Positive** is the value returned the calle or Parent.

- **Waitpid**
  - Syntax: pid_t waitpid (child_pid, &status, options);
  - It returns the pid of child program was waiting.
  - It uses int* to describe the status of the child.
  - And It has options like WNOHANG.

Error Handling

- Handled missing file in open
- Handled eof in read
- Handled negative values in fork