Q1 :
code ;

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

int main() {
	int fd1[2], fd2[2] ;
	pipe(fd1) ;
	pipe(fd2) ;
	if( fork() == 0){
		close(fd1[1]) ;
		char buf[100];
		read(fd1[0], buf , sizeof(buf)) ;
		printf("The child reads : %s \n", buf) ;
		char ans[100] ;
		int len = 0 ;
		for(int i = 0 ; buf[i]!='\0'; i++ ) {
			char curr = buf[i] ;
			len++ ;
			char newc ;
			if((int)curr > 90){
				newc = toupper(curr) ;
			}
			else{
				newc = tolower(curr) ;
			}

			ans[i] = newc ;
		}
		ans[len] = '\0' ;
		close(fd2[0]) ;
		write(fd2[1], ans , len + 1) ;
		close(fd2[1]) ;
	}
	else{
		close(fd1[0]) ;
		char mess[100] ;
		scanf("%99[^\n]",mess) ;
		write(fd1[1],mess , strlen(mess)+1) ;
		close(fd1[1]) ;

		close(fd2[1]) ;
		char buff[100]  ;
		read(fd2[0] , buff , sizeof(buff) ) ;
		printf("this is what the parent recieves from the child: %s \n",buff) ;
```

```
        close(fd2[0]) ;

    }

}
```

```
cse@cse:~/os/Lab4$ ./q1
hel000
The child reads : hel000
this is what the parent recieves from the child: HELooo
```

q2A :

```
// author : bibhuti
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#include <sys/wait.h>

int main(int argc , char *argv[]){
    if( argc != 2) {
        printf("Usage: %s <num_terms>\n", argv[0]) ;
        return 1 ;
    }
    int num = atoi(argv[1]) ;
    if(num <= 0) {
        printf("Error , the number entered cannt be negative") ;
        return(1) ;
    }
    if(fork() == 0 ) {
        //child it is
        int n = num ;
        int dp[n+2] ;
        dp[0]= 0 ;
        dp[1]= 1;
        dp[2]= 1 ;
        for(int i = 3 ; i<= n ; i++ ) {
            dp[i] = dp[i-1] + dp[i-2] ;
        }
        for(int i = 0; i< n ; i++ ) {
            printf("%d ",dp[i]) ;
        }
        printf("\n") ;
        return(0) ;

    }
    else{
        //parent :
```

```
        wait(NULL) ;
        printf("Finally parents end after child finished") ;
        return(0) ;
        //waits for the child to finish
    }
}
~
```

```
cse@cse:~/os/Lab4$ vim q2.c
cse@cse:~/os/Lab4$ gcc q2.c -o q2
cse@cse:~/os/Lab4$ ./q2 5
0 1 1 2 3
Finally parents end after child finishedcse@cse:~
```

q2B

```
// author : bibhuti
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <ctype.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/wait.h>

#define MAX 10
typedef struct{
    long fib_seq[MAX] ;
    int seq_size ;
} shared_data ;

int main(int argc , char *argv[]){
    if( argc != 2) {
        printf("Usage: %s <num_terms>\n", argv[0]) ;
        return 1 ;
    }
    int num = atoi(argv[1]) ;
    if(num <= 0) {
        printf("Error , the number entered cannt be negative") ;
        return(1) ;
    }

    if(num > MAX ) {
        printf(" Error by parent , cauz the size breaks the limits  \n") ;
        return(1) ;
    }
```

```
	const char *name = "/mysharedmemory";
	int sz = sizeof(shared_data) ;
	int f = shm_open(name , O_CREAT | O_RDWR , 0666) ;
	ftruncate(f, sz) ;
	shared_data *shm_pointer = mmap(0 , sz , PROT_READ | PROT_WRITE ,
MAP_SHARED, f, 0);
	if( shm_pointer == MAP_FAILED ) {
		perror("mmap") ;
		exit(1) ;
	}

	if(fork() == 0 ) {
		//child it is
		shm_pointer->seq_size = num ;
		shm_pointer->fib_seq[0]= 0 ;
		if(num >1) shm_pointer->fib_seq[1]= 1;
		for(int i = 2; i< num ; i++ ) {
			shm_pointer->fib_seq[i] =
shm_pointer->fib_seq[i-1]+shm_pointer->fib_seq[i-2] ;
		}

		munmap(shm_pointer , sz ) ;
		close(f) ;
		return(0) ;

	}
	else{
		//parent :

		wait(NULL) ;
		printf("the wait for the child to do its operation is finished , now lets process .. \n") ;
		printf("here is the fibonacci sequence \n") ;
		for(int i  = 0 ; i< shm_pointer->seq_size  ; ++i ) {
			printf("%ld ", shm_pointer->fib_seq[i]) ;
		}
		printf("\n") ;

		munmap(shm_pointer , sz) ;
		close(f) ;
		shm_unlink(name) ;

		return(0) ;
		//waits for the child to finish
	}
}


~
```

~

```
cse@cse:~/os/Lab4$ gcc q2b.c -o q2b -lrt
cse@cse:~/os/Lab4$ ./q2b 11
 Error by parent , cauz the size breaks the limits
cse@cse:~/os/Lab4$ ./q2b 9
the wait for the child to do its operation is finished , now lets process ..
here is the fibonacci sequence
0 1 1 2 3 5 8 13 21
cse@cse:~/os/Lab4$
```

q3 :

I was able to do the editor.c portion only , not proficient in threads yet

journalist is left still , couldnt do it

```
// author : bibhuti
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <ctype.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/wait.h>

int main(int argc , char *argv[]) {
	if(argc != 3 ) {
		printf("not correct command \n") ;
		return(1) ;
	}

	int N = atoi(argv[1]) ;
	int M = atoi(argv[2]) ;

	int fd[N][2] ;
	for(int i = 0 ; i<N ; i++ ){
		pipe(fd[i]) ;
	}
	pid_t pids[N] ;
	for(int i = 0 ; i<N ; i++ ) {
```

```
        pid_t pid = fork() ;
        if( pid == 0) {
            close(fd[i][0]) ; //closing the read end of that pipe cauz ye child will write
            char id_str[10] , fd_str[10] ;
            sprintf(id_str , "%d", i+1) ;
            sprintf(fd_str , "%d", df[i][1]) ; //write end of the pipe we just send

            execlp("./journalist", "journalist" , id_str , fd_str , NULL) ;
            perror("execlp") ;
            exit(1) ;
        }
        else {
            //parent stuff
            close(fd[i][1]) ;
            pids[i]= pid ;
        }


    }
    int counter = 0 ;
    while(counter < M) {
        fd_set readfds ;
        FD_ZERO(&readfds) ;
        int maxfd = -1 ;
        for(int i = 0 ; i< N ; i++ ) {
            if(FD_ISSET(fd[i][0], &readfds)) {
                char buffer[256];
                int bytes = read(fd[i][0] , buffer , sizeof(buffer)-1) ;
                if (bytes > 0) {
                    buffer[bytes] = '\0';
                    printf("Editor: received article from Journalist %d: %s\n", i + 1, buffer);
                    counter++;
                    if (counter >= M)
                        break;
                }
            }
        }
    }
    printf("The editor got all the articles from the jounralists total = %d \n",M);
    for(int i = 0 ; i<N ; i++ ){
        kill(pids[i] , SIGTERM) ;
    } // we killed all the processes
    for (int i = 0; i < N; i++) {
        close(fd[i][0]);
        waitpid(pids[i], NULL, 0);
    }

printf("Editor: All journalists terminated. Exiting.\n");
```

```
    return 0;

}
```