1.

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/errno.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
   int fd;
   char read_byte, input_pattern = 'a', output_pattern = '$';

   if (argc != 2)
   {

      write(STDOUT_FILENO, 'less arguments', 14);
      exit(0);
   }
   else
   {

      fd = open(argv[1], O_RDWR);
      if (fd != -1)
      {
         while (read(fd, &read_byte, 1) > 0)
```

```
        {
            if (read_byte == input_pattern)
            {
                lseek(fd, -1, SEEK_CUR);
                write(fd, &output_pattern, sizeof(output_pattern));
            }

        }
        close(fd);

    }
    else
    {

        write(STDOUT_FILENO, &errno, sizeof(errno));
        perror("File Open");
        exit(0);
    }
  }

  return 0;
}
```

2.

```
#include <stdio.h>
#include <sys/types.h>
```

```
#include<wait.h>
#include<unistd.h>
#include <dirent.h>
#include <stdlib.h>
#include<sys/errno.h>

void show_content(){
   DIR *d;
   struct dirent *dir;
   d = opendir(".");
   if (d)
   {
      printf("\nContent of the current directory goes here...");
      while ((dir = readdir(d)) != NULL)
      {
         printf("\n%s", dir->d_name);
      }
      closedir(d);
   }
}

int main(int argc, char *argv[])
{

   pid_t pid;
```

```
    int status;


    pid = fork();
    if (pid == 0)
    {
        printf("\n Entered in child...");
        show_content();
        printf("\n child finishes execution...");


    }else if (pid > 0)
    {
        wait(&status);
        printf("\nEntered in parent...");
        printf("\nParent process finishes execution...");
    }else
    {
        perror("Fork Open failed");
        exit(errno);
    }
    return 0;
}
```