The odds

Write a program that reads a file and sends the contents of a file to the child. Then child prints the first five characters and then terminates. Parent process must terminate after the child and print the statuses when creating the child process and upon the completion of the child process. Assume the file has more than five characters.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<string.h>
#include<sys/wait.h>

int main()
{

    int pipefd[2];
    char fileName[] = “myfile.txt”;
    pid_t pid;
    if (pipe2(pipefd,O_NONBLOCK)==-1) // to avoid blocking call
    {
        fprintf(stderr, "Pipe Failed" );
        return 1;
    }
    printf(“creating child”);
    pid = fork();
    if (pid < 0)
    {
        fprintf(stderr, "fork Failed" );
        return 1;
    }
    else if (pid > 0)
    {
        printf(“child created”);
        close(pipefd[0]); // Close reading end of first pipe
        FILE* fd = open(fileName,”+r”);
        char c = 0;
```

```
        while (read(fd,&c,1))
            write(pipefd[1],c, 1);
        wait(NULL);
        printf("child terminated");
    }

    // child process
    else
    {
        printf("child started");
        close(pipefd[1]); // Close writing end of first pipe
        int i = 5;
        while (i > 0){
            char c = 0;
            read(pipefd[0],&c,1);
            printf("%c", c);
            --i;
        }
        exit(0);
    }
}
```

The evens

Write a program that creates a child process. The child process reads input from the user and after reading five inputs the child sends the inputs to parent and then terminates. Parent process shows the data read from the child. Parent process must terminate after the child and print the statuses when creating the child process and upon the completion of the child process

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<string.h>
#include<sys/wait.h>

int main()
```

```
{

    int pipefd[2];
    pid_t pid;
    if (pipe(pipefd)==-1)
    {
        fprintf(stderr, "Pipe Failed" );
        return 1;
    }
    printf(“creating child”);
    pid = fork();
    if (pid < 0)
    {
        fprintf(stderr, "fork Failed" );
        return 1;
    }
    else if (pid > 0)
    {
        printf(“child created”);
        close(pipefd[1]); // Close writing end of first pipe
        char c = 0;
        while (read(pipefd,&c,1))
            printf(“%c”,c);
        wait(NULL);
        printf(“child terminated”);
    }

    // child process
    else
    {
        printf(“child started”);
        close(pipefd[0]); // Close reading end of first pipe
        int i = 5;
        while (i > 0){
            char c = 0;
            scanf(“%c”,&c);
            write(pipefd[0],&c,1);
            --i;
        }
        exit(0);
    }
}
```