13.Write a C program that behaves like a shell (command interpreter). It has its

own prompt say “NewShell$”. Any normal shell command is executed from

your shell by starting a child process to execute the system program

corresponding to the command. It should additionally interpret the following

command.

i) typeline +10 <filename> - print first 10 lines of file

ii) typeline -20 <filename> - print last 20 lines of file

iii) typeline a <filename> - print all lines of file

#include<stdio.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<unistd.h>

#include<dirent.h>

#include<fcntl.h>

void typeline(char \*s, char \*fn)

{

int handle,i=0,cnt=0,n;

char ch;

if((handle=open(fn,O\_RDONLY))==-1)

{

printf("File %s not found\n",fn);

return;

}

if(strcmp(s,"a")==0)

{

while(read(handle,&ch,1)!=0)

printf("%c",ch);

close(handle);

return;

}

n=atoi(s);

if(n>0)

{

while(read(handle,&ch,1)!=0)

{

if(ch=='\n')

i++;

if(i==n)

break;

printf("%c",ch);

}

printf("\n");

close(handle);

return;

}

if(n<0)

{

while(read(handle,&ch,1)!=0)

{

if(ch=='\n')

cnt++;

}

/\*lseek is used to change location of read and write pointer of a file descriptor\*/

lseek(handle,0,SEEK\_SET);

while(read(handle,&ch,1)!=0)

{

if(ch=='\n')

i++;

if(i==((cnt-1)+n-1))

break;

}

while(read(handle,&ch,1)!=0)

printf("%c",ch);

printf("\n");

close(handle);

}

}

main()

{

char command[80],t1[20],t2[20],t3[20],t4[20];

int n;

system("clear");

while(1)

{

printf("NewShell$");

fflush(stdin);

fgets(command,80,stdin);

n = sscanf(command,"%s %s %s %s",t1,t2,t3,t4);

switch(n)

{

case 1:

if(!fork())

{

execlp(t1,t1,NULL);

perror(t1);

}

break;

case 2:

if(!fork())

{

execlp(t1,t1,t2,NULL);

perror(t1);

}

break;

case 3:

if(strcmp(t1,"typeline")==0)

typeline(t2,t3);

else

{

if(!fork())

{

execlp(t1,t1,t2,t3,NULL);

perror(t1);

}

}

break;

case 4:

if(!fork())

{

execlp(t1,t1,t2,t3,t4,NULL);

perror(t1);

}

}

}

}