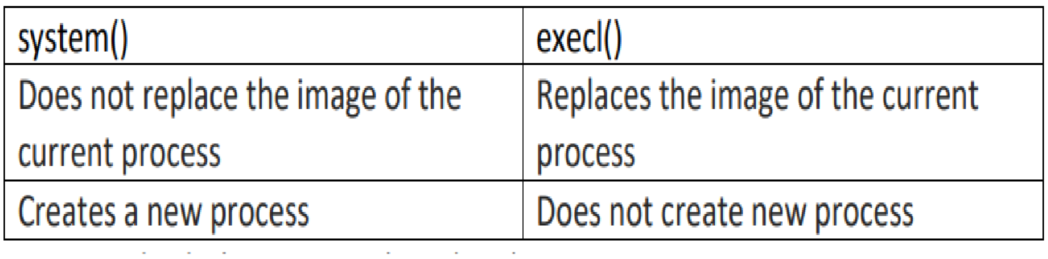
Name:	VENNELA G
Register No:	20BDS0146
Lab Course Name:	OPERATING SYSTEMS
Lab Slot:	L21+L22
Lab Assessment Title:	PROCESS MANAGEMENT

Question 1

Study of differences between system( ) and execl( ) /execlp( ) calls. Give examples to run user defined programs and OS commands using system( ) and execl( ) /execlp( ) calls.

Answer:



system()	execl()
Does not replace the image of the current process	Replaces the image of the current process
Creates a new process	Does not create new process

system()

system() function is used to execute a shell command from within a process.

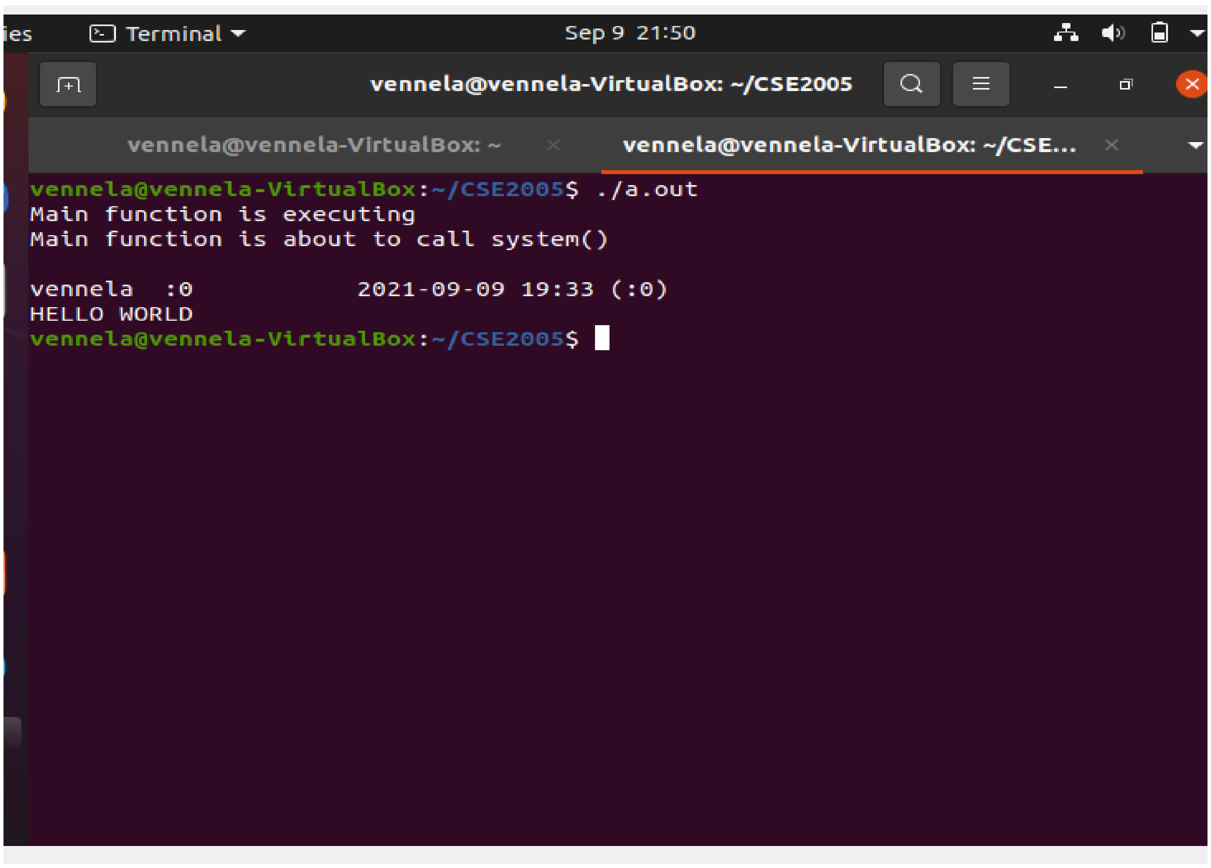
Syntax of system():

```
#include
 int system(const char *command);
```

SOURCE CODE:

```
#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>


int main()

{printf("Main function is executing\n");

printf("Main function is about to call system()\n\n");

system("who");

printf("HELLO WORLD\n");

 return 0;

}
```

OUTPUT:


ies
Terminal
Sep 9 21:50
vennela@vennela-VirtualBox: ~/CSE2005
vennela@vennela-VirtualBox: ~
vennela@vennela-VirtualBox: ~/CSE...

```
vennela@vennela-VirtualBox:~/CSE2005$ ./a.out
Main function is executing
Main function is about to call system()

vennela  :0           2021-09-09 19:33 (:0)
HELLO WORLD
vennela@vennela-VirtualBox:~/CSE2005$
```

execlp()

execl() functions replace the image of the current process with a new process image.

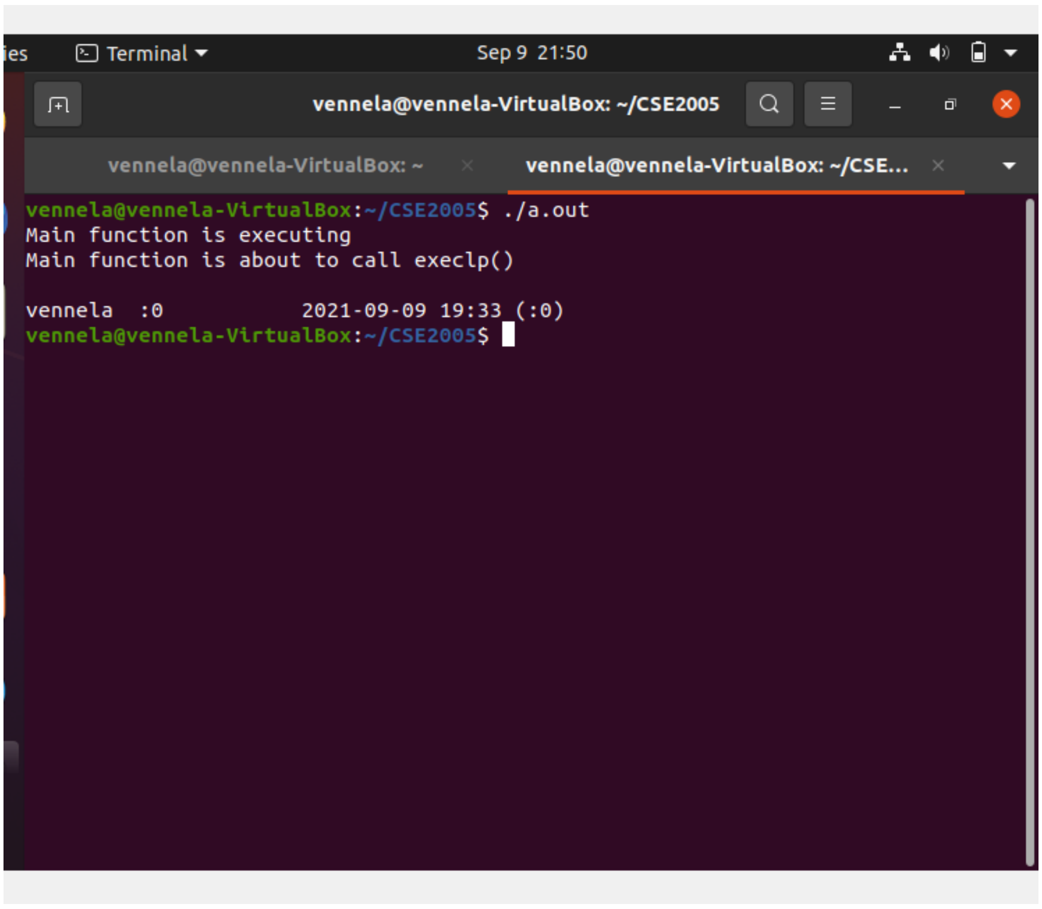
Syntax of execlp():

```
#include

int execl(const char *path, const char *arg, . . . /* (char *) NULL */);
```

SOURCE CODE:

```
#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>


int main()

{printf("Main function is executing\n");

printf("Main function is about to call execlp()\n\n");

execlp("/usr/bin/who","who",NULL);

printf("HELLO WORLD\n");

 return 0;

}
```

OUTPUT:


ies
Terminal
Sep 9 21:50
vennela@vennela-VirtualBox: ~/CSE2005
vennela@vennela-VirtualBox: ~
vennela@vennela-VirtualBox: ~/CSE...

```
vennela@vennela-VirtualBox:~/CSE2005$ ./a.out
Main function is executing
Main function is about to call execlp()

vennela  :0           2021-09-09 19:33 (:0)
vennela@vennela-VirtualBox:~/CSE2005$ 
```

Question 2

Write a C program to create a child process.

- Let the child process be assigned the task of checking if two input strings are Isomorphic strings.
- Let the parent be checking if two input strings are Anagrams.

SOURCE CODE:

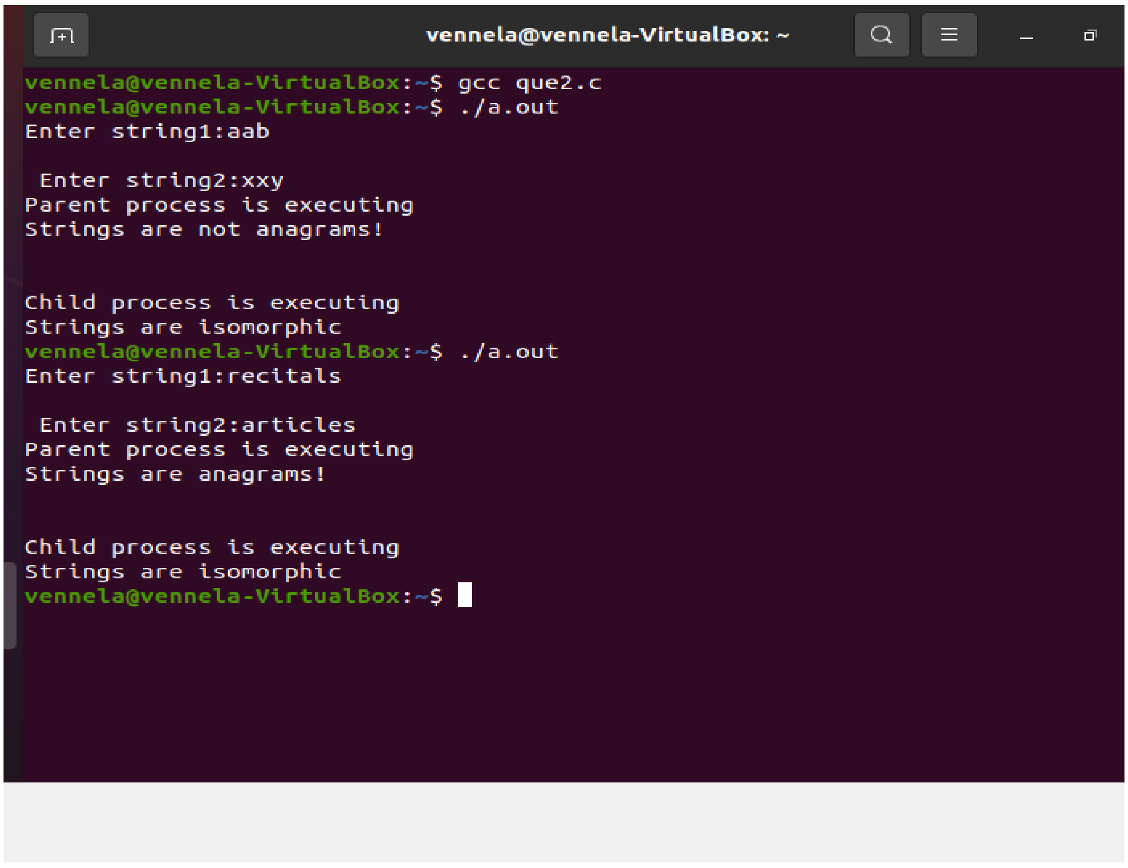
```
#include<unistd.h>

#include<stdio.h>

#include<sys/types.h>

#include<string.h>


int main()

{pid_t pid;

char s1[50],s2[50],buf[50];

int arr[256]={0},arr1[256]={0};

printf("Enter string1:");

scanf("%[^\n]s",s1);

printf("\n Enter string2:");

fgets(buf,50,stdin);

scanf("%[^\n]s",s2);

fflush(stdout);

pid=fork();

if(pid<0){


perror("fork");

return 0;
```

```
}
if(pid==0){//child executes here
printf("\n\nChild process is executing\n");
 if( strlen(s1) != strlen(s2)) {
      printf("Strings are not isomorphic \n");
     return 0;
   }
 for (int i = 0; i < strlen(s1); i++) {
            if (arr[(int)s1[i]]
        != arr1[(int)s2[i]]) {
            printf("Strings are not isomorphic \n");
            return 0;

        }



        arr[(int)s1[i]]++;
        arr1[(int)s2[i]]++;
    }
    printf("Strings are isomorphic \n");
     return 0;

}



else{//parent executes here
printf("Parent process is executing\n");
char temp;
```

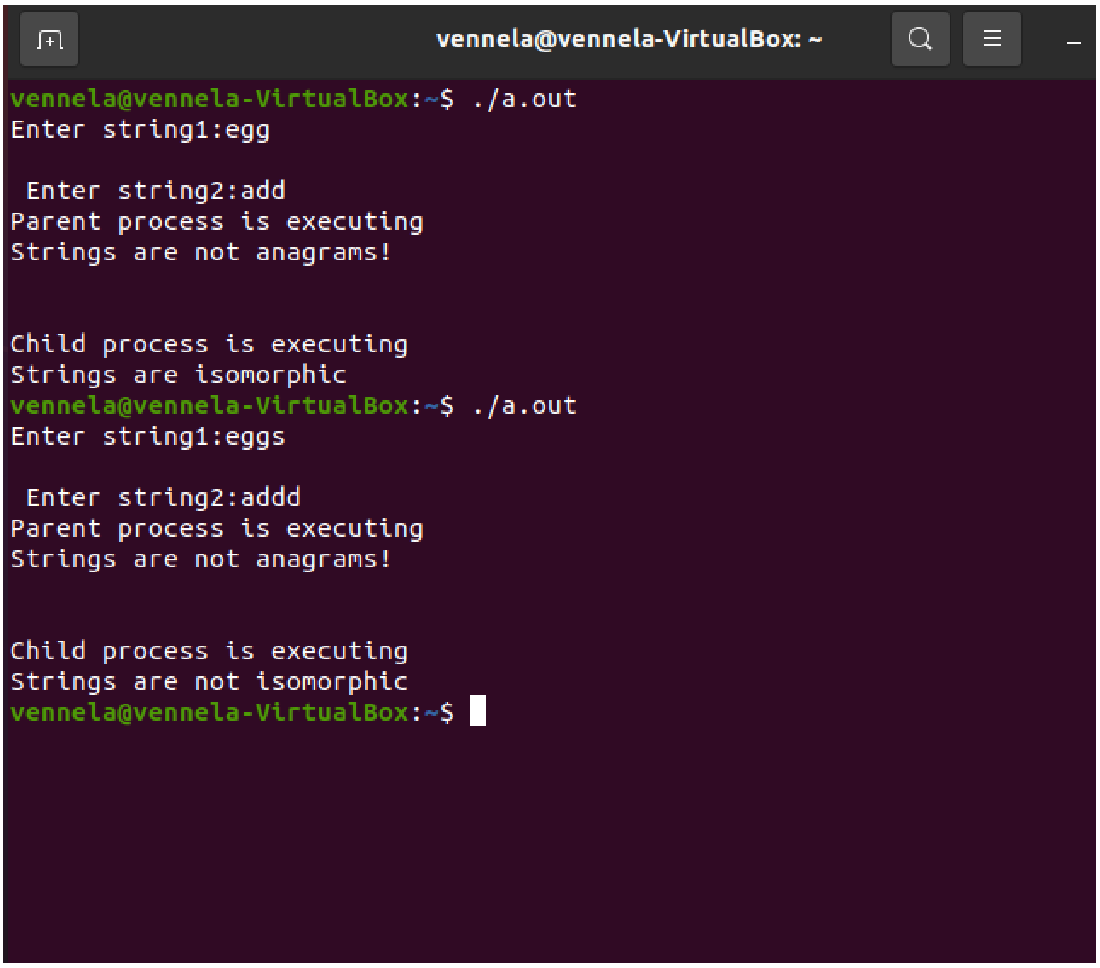
```
   int i, j;

   int n  = strlen(s1);

   int n1 = strlen(s2);

if( n != n1) {

      printf("Strings are not anagrams! \n");

     return 0;

   }



   for (i = 0; i < n-1; i++) {

      for (j = i+1; j < n; j++) {

         if (s1[i] > s1[j]) {

            temp  = s1[i];

            s1[i] = s1[j];

            s1[j] = temp;

         }

         if (s2[i] > s2[j]) {

            temp  = s2[i];

            s2[i] = s2[j];

            s2[j] = temp;

         }

      }

   }




   for(i = 0; i<n; i++) {

      if(s1[i] != s2[i]) {

         printf("Strings are not anagrams! \n");
```

```
        return 0;
}
   }

   printf("Strings are anagrams! \n");
  return 0;
}

return 0;
}
```

OUTPUT:



```
vennela@vennela-VirtualBox:~$ gcc que2.c
vennela@vennela-VirtualBox:~$ ./a.out
Enter string1:aab

 Enter string2:xxy
Parent process is executing
Strings are not anagrams!


Child process is executing
Strings are isomorphic
vennela@vennela-VirtualBox:~$ ./a.out
Enter string1:recitals

 Enter string2:articles
Parent process is executing
Strings are anagrams!


Child process is executing
Strings are isomorphic
vennela@vennela-VirtualBox:~$ 
```


vennela@vennela-VirtualBox: ~

```
vennela@vennela-VirtualBox:~$ ./a.out
Enter string1:egg

 Enter string2:add
Parent process is executing
Strings are not anagrams!


Child process is executing
Strings are isomorphic
vennela@vennela-VirtualBox:~$ ./a.out
Enter string1:eggs

 Enter string2:addd
Parent process is executing
Strings are not anagrams!


Child process is executing
Strings are not isomorphic
vennela@vennela-VirtualBox:~$ 
```

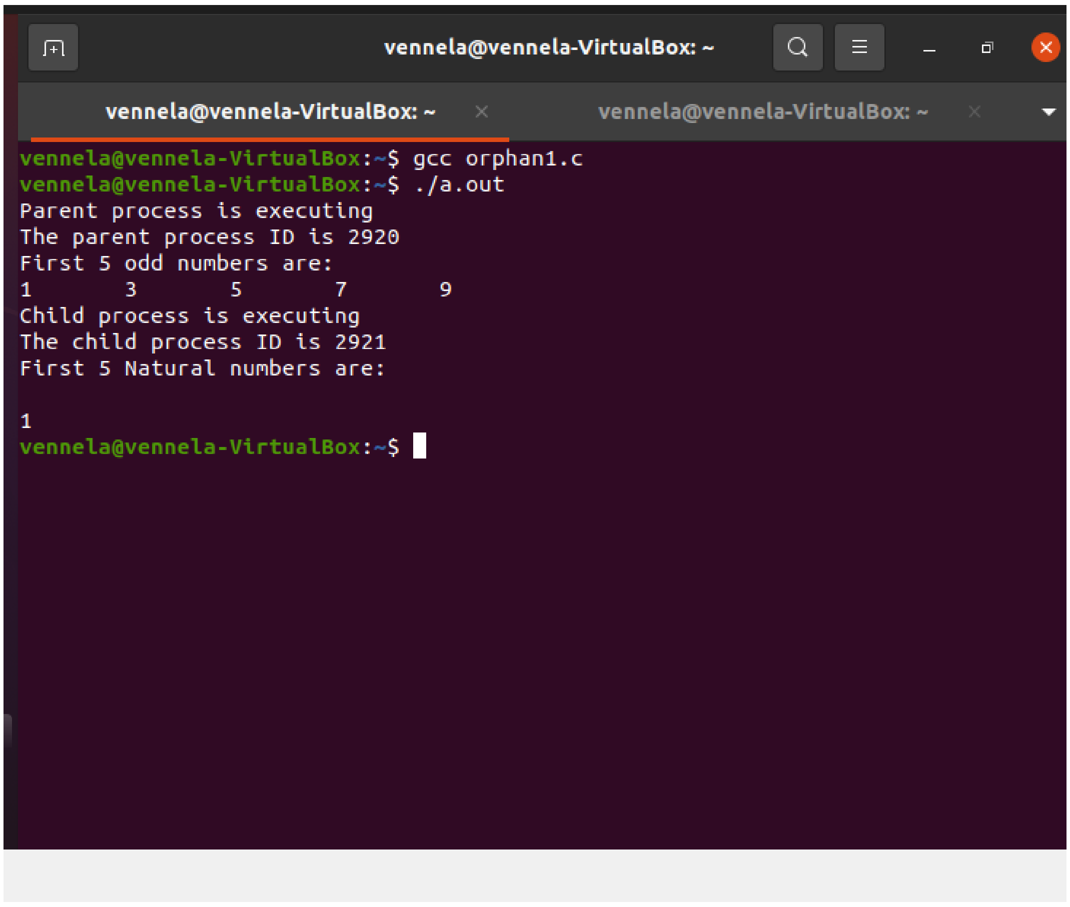
Question 3

Write a C Program to create an Orphan process (Do not use the same approach as discussed in the class).

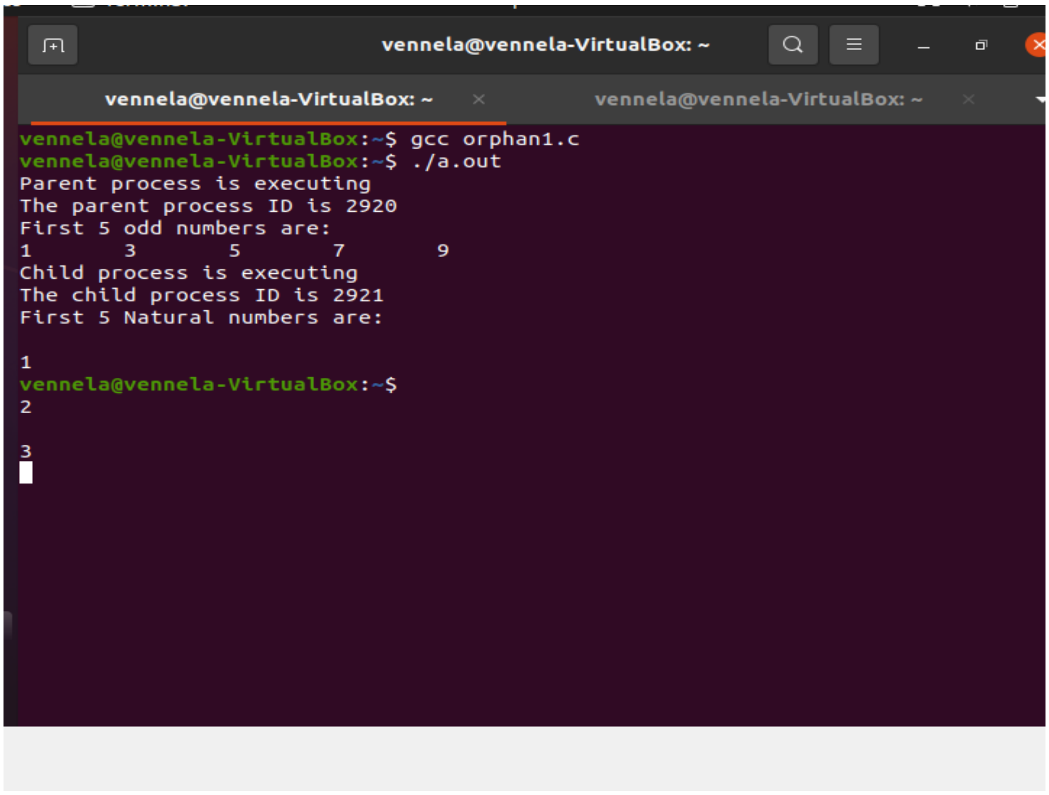
SOURCE CODE:

```
#include<unistd.h>

#include<stdio.h>

#include<sys/types.h>

int  main()

{pid_t pid;int k,j;

pid=fork();

int i=0;

if(pid<0){


perror("fork");

return 0;

}

if(pid>0){//parent executes here

printf("Parent process is executing\n");

printf("The parent process ID is %u\n",getpid());;

printf("First 5 odd numbers are:\n");

for(k=1;k<10;k+=2)

 {printf("%d\t",k);}}


else if(pid==0){//child executes here

printf("\nChild process is executing\n");

printf("The child process ID is %u\n",getpid());

printf("First 5 Natural numbers are:\n");

 for(j=1;j<6 ;j++)
 { printf("\n%d\n",j);
    sleep(5);}}
return 0;
}
```

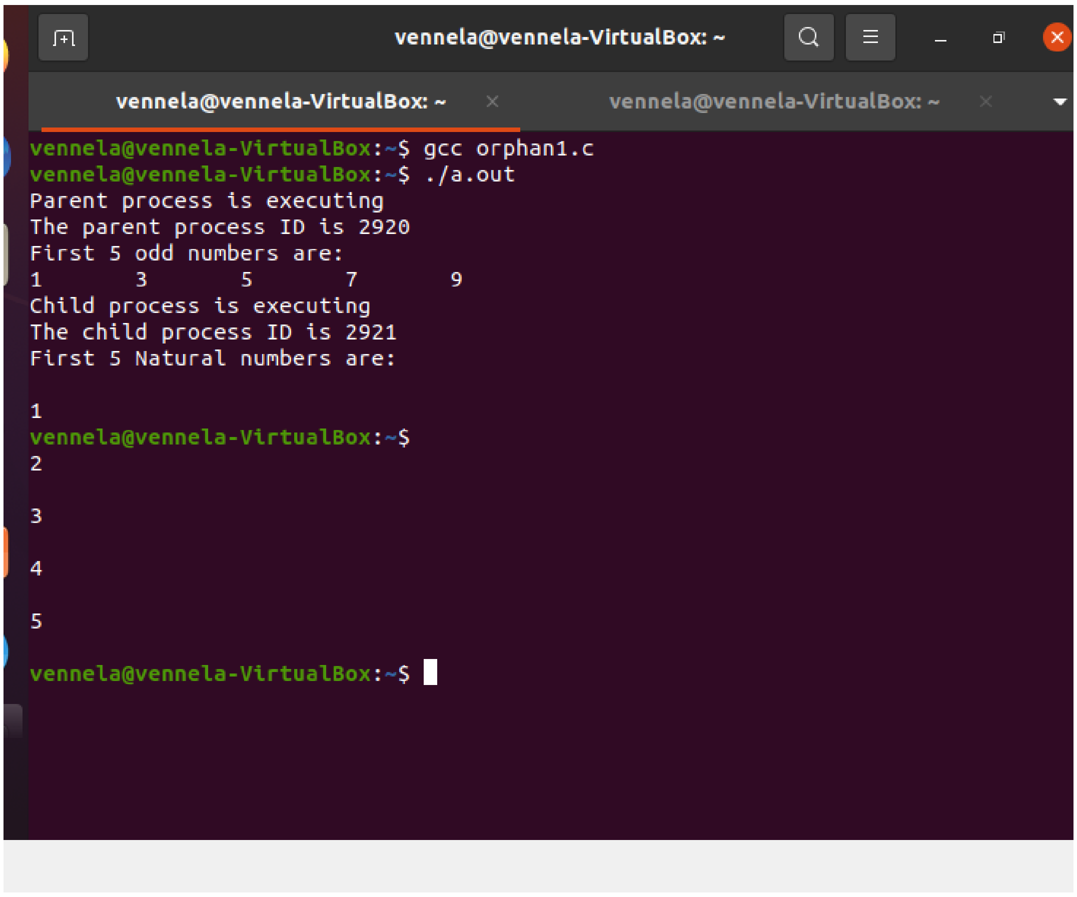
OUTPUT:



```
vennela@vennela-VirtualBox:~$ gcc orphan1.c
vennela@vennela-VirtualBox:~$ ./a.out
Parent process is executing
The parent process ID is 2920
First 5 odd numbers are:
1       3       5       7       9
Child process is executing
The child process ID is 2921
First 5 Natural numbers are:

1
vennela@vennela-VirtualBox:~$
```



```
vennela@vennela-VirtualBox:~$ gcc orphan1.c
vennela@vennela-VirtualBox:~$ ./a.out
Parent process is executing
The parent process ID is 2920
First 5 odd numbers are:
1       3       5       7       9
Child process is executing
The child process ID is 2921
First 5 Natural numbers are:

1
vennela@vennela-VirtualBox:~$
2

3
```



```
vennela@vennela-VirtualBox:~$ gcc orphan1.c
vennela@vennela-VirtualBox:~$ ./a.out
Parent process is executing
The parent process ID is 2920
First 5 odd numbers are:
1       3       5       7       9
Child process is executing
The child process ID is 2921
First 5 Natural numbers are:

1
vennela@vennela-VirtualBox:~$
2

3

4

5

vennela@vennela-VirtualBox:~$
```

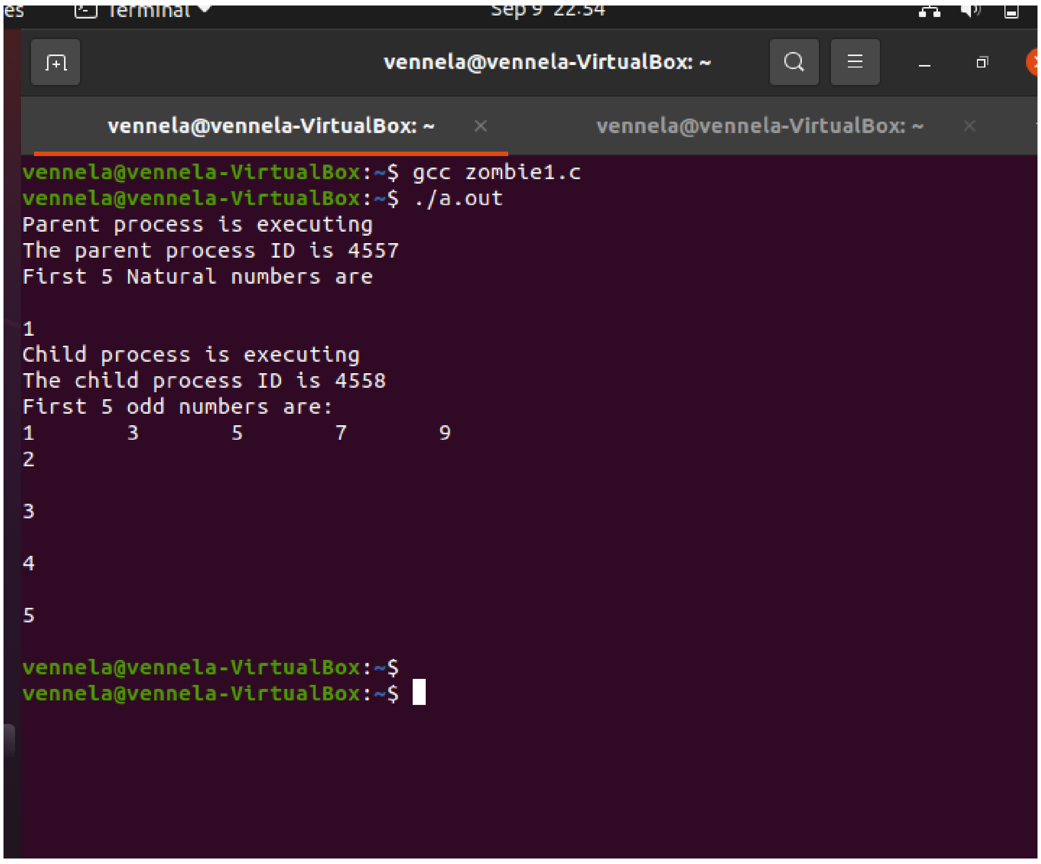
Question 4

Write a C Program to create a Zombie process (Do not use the same approach as discussed in the class).

SOURCE CODE:

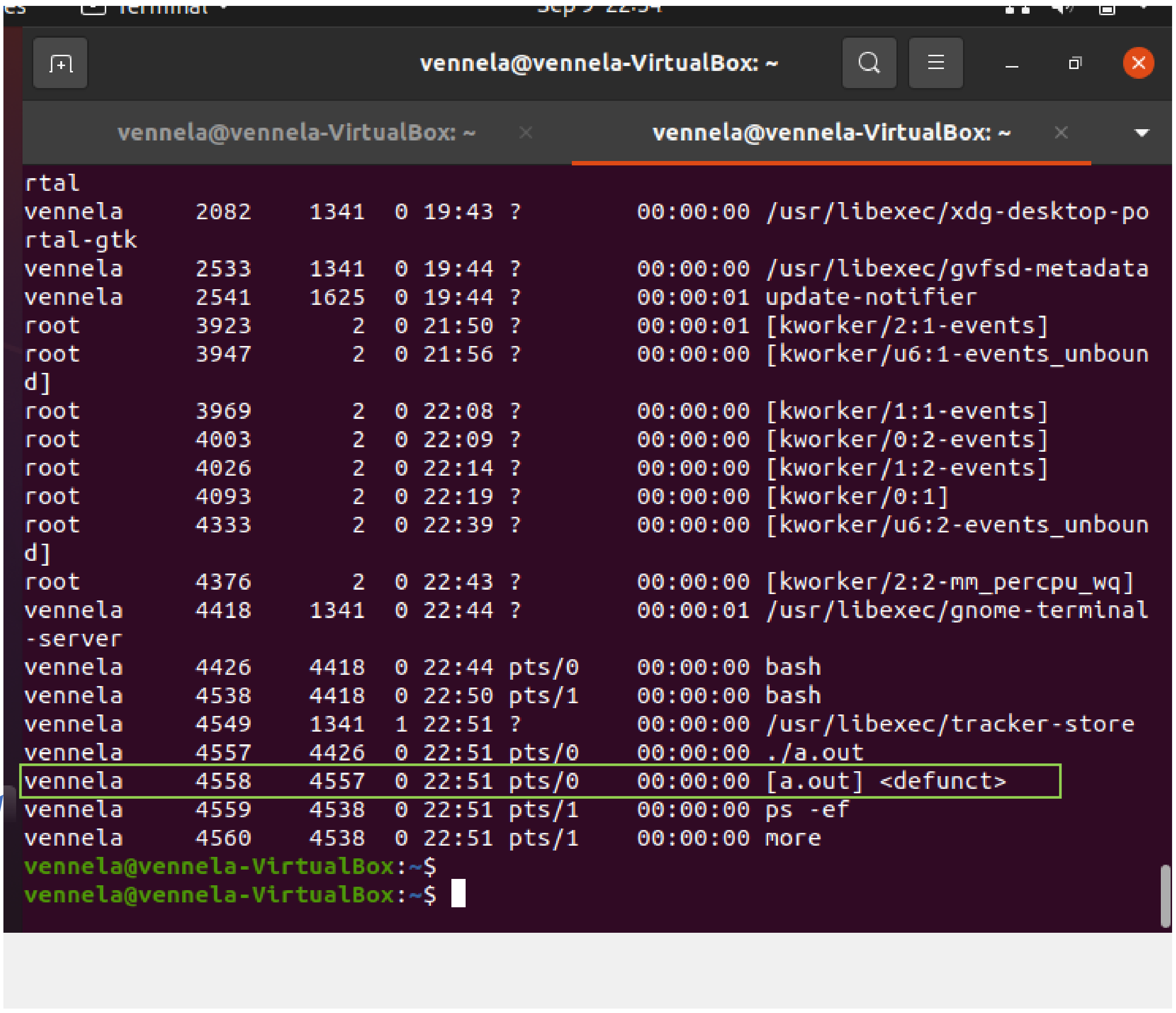
```
#include<unistd.h>

#include<stdio.h>

#include<sys/types.h>

int main()

{pid_t pid;int k,j;

pid=fork();

int i=0;

if(pid<0){


perror("fork");

return 0;

}

if(pid==0){//child executes here

printf("Child process is executing\n");

printf("The child process ID is %u\n",getpid());

printf("First 5 odd numbers are:\n");

for(k=1;k<10;k+=2)

 {printf("%d\t",k);}

}

else{//parent executes here

printf("Parent process is executing\n");

printf("The parent process ID is %u\n",getpid());;

 printf("First 5 Natural numbers are\n");

 for(j=1;j<6 ;j++)
 { printf("\n%d\n",j);
       sleep(5);


}}
return 0;
}
```

OUTPUT:



Terminal
Sep 9 22:54
vennela@vennela-VirtualBox: ~
vennela@vennela-VirtualBox: ~
vennela@vennela-VirtualBox: ~

```
vennela@vennela-VirtualBox:~$ gcc zombie1.c
vennela@vennela-VirtualBox:~$ ./a.out
Parent process is executing
The parent process ID is 4557
First 5 Natural numbers are

1
Child process is executing
The child process ID is 4558
First 5 odd numbers are:
1       3       5       7       9
2

3

4

5

vennela@vennela-VirtualBox:~$
vennela@vennela-VirtualBox:~$ 
```


vennela@vennela-VirtualBox: ~
vennela@vennela-VirtualBox: ~
vennela@vennela-VirtualBox: ~

```
rtal
vennela     2082    1341  0 19:43 ?        00:00:00 /usr/libexec/xdg-desktop-po
rtal-gtk
vennela     2533    1341  0 19:44 ?        00:00:00 /usr/libexec/gvfsd-metadata
vennela     2541    1625  0 19:44 ?        00:00:01 update-notifier
root        3923       2  0 21:50 ?        00:00:01 [kworker/2:1-events]
root        3947       2  0 21:56 ?        00:00:00 [kworker/u6:1-events_unboun
d]
root        3969       2  0 22:08 ?        00:00:00 [kworker/1:1-events]
root        4003       2  0 22:09 ?        00:00:00 [kworker/0:2-events]
root        4026       2  0 22:14 ?        00:00:00 [kworker/1:2-events]
root        4093       2  0 22:19 ?        00:00:00 [kworker/0:1]
root        4333       2  0 22:39 ?        00:00:00 [kworker/u6:2-events_unboun
d]
root        4376       2  0 22:43 ?        00:00:00 [kworker/2:2-mm_percpu_wq]
vennela     4418    1341  0 22:44 ?        00:00:01 /usr/libexec/gnome-terminal
-server
vennela     4426    4418  0 22:44 pts/0    00:00:00 bash
vennela     4538    4418  0 22:50 pts/1    00:00:00 bash
vennela     4549    1341  1 22:51 ?        00:00:00 /usr/libexec/tracker-store
vennela     4557    4426  0 22:51 pts/0    00:00:00 ./a.out
vennela     4558    4557  0 22:51 pts/0    00:00:00 [a.out] <defunct>
vennela     4559    4538  0 22:51 pts/1    00:00:00 ps -ef
vennela     4560    4538  0 22:51 pts/1    00:00:00 more
vennela@vennela-VirtualBox:~$
vennela@vennela-VirtualBox:~$
```

Zombie process

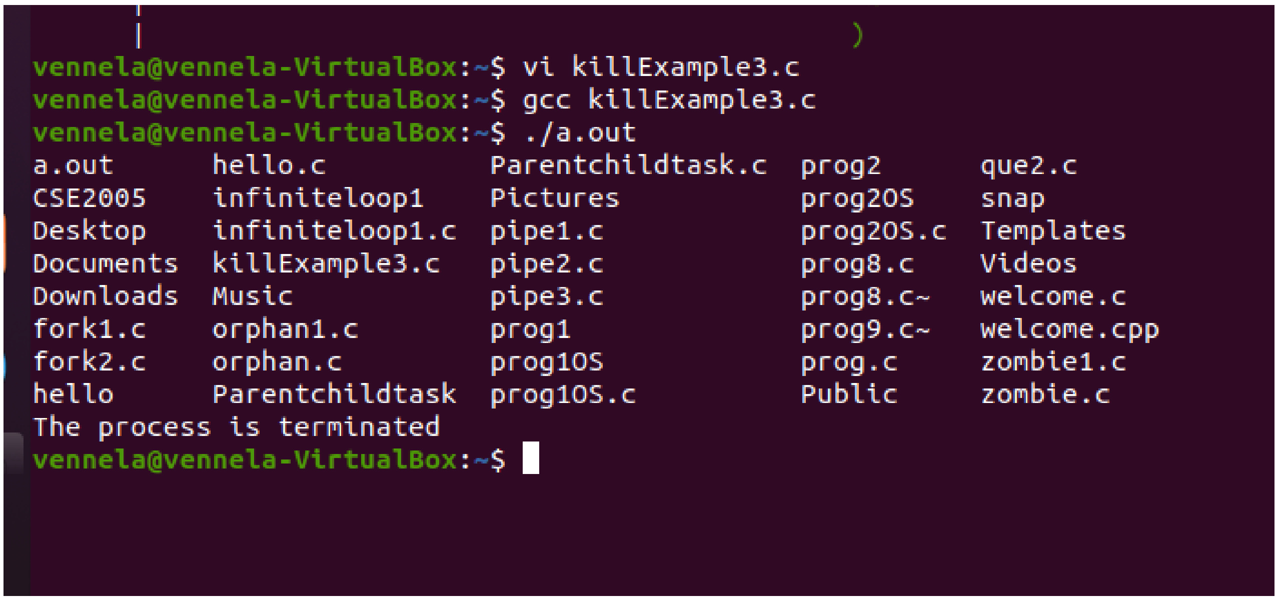
Question 5

Write a C program to kill a process given its name.

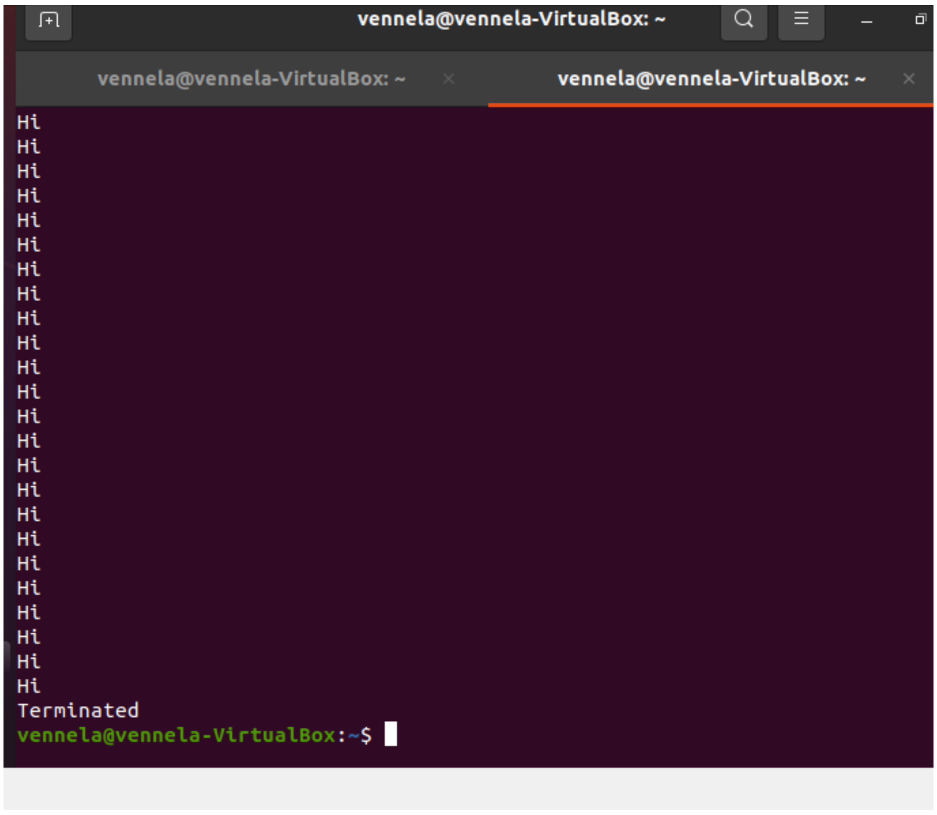
SOURCE CODE:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>


int main()

{


  char die[BUFSIZ];

  char child[BUFSIZ];

  int ans;



ans=system("/usr/bin/ls");



if(ans==-1){

        return 0;

}



while(ans != 0);

 {

        strcpy (die, "killall infiniteloop1\n");

        system(die);
        printf("The process is terminated\n");
  }
return 0;
}
```

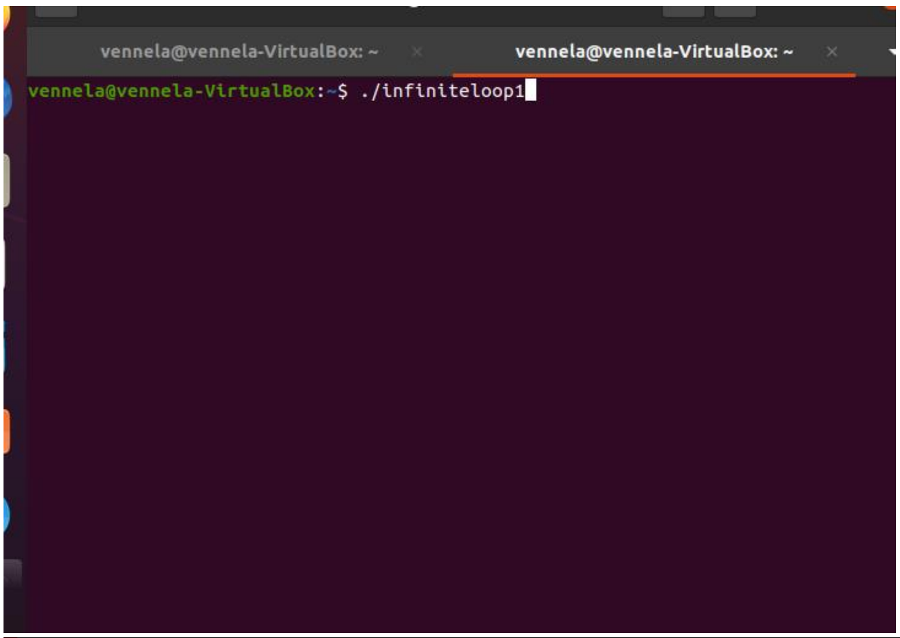
OUTPUT:



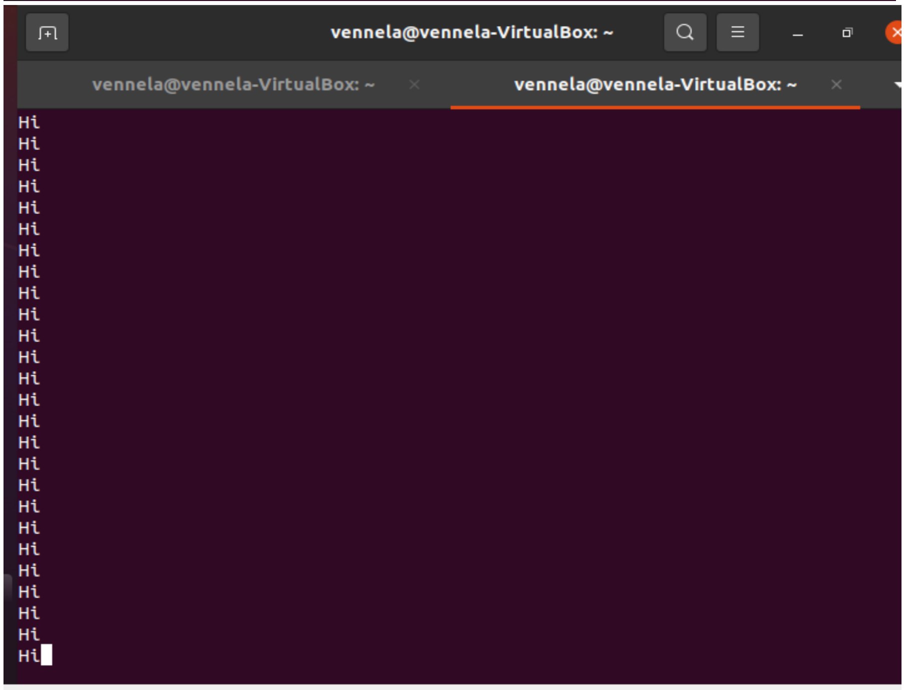
```
vennela@vennela-VirtualBox:~$ vi killExample3.c
vennela@vennela-VirtualBox:~$ gcc killExample3.c
vennela@vennela-VirtualBox:~$ ./a.out
a.out      hello.c          Parentchildtask.c   prog2      que2.c
CSE2005    infiniteloop1    Pictures            prog2OS    snap
Desktop    infiniteloop1.c  pipe1.c             prog2OS.c  Templates
Documents  killExample3.c   pipe2.c             prog8.c    Videos
Downloads  Music            pipe3.c             prog8.c~   welcome.c
fork1.c    orphan1.c        prog1               prog9.c~   welcome.cpp
fork2.c    orphan.c         prog1OS             prog.c     zombie1.c
hello      Parentchildtask  prog1OS.c           Public     zombie.c
The process is terminated
vennela@vennela-VirtualBox:~$ 
```


vennela@vennela-VirtualBox: ~
vennela@vennela-VirtualBox: ~
vennela@vennela-VirtualBox: ~

```
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Terminated
vennela@vennela-VirtualBox:~$
```



```
vennela@vennela-VirtualBox:~$ ./infiniteloop1
```


vennela@vennela-VirtualBox: ~
vennela@vennela-VirtualBox: ~
vennela@vennela-VirtualBox: ~

```
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
```