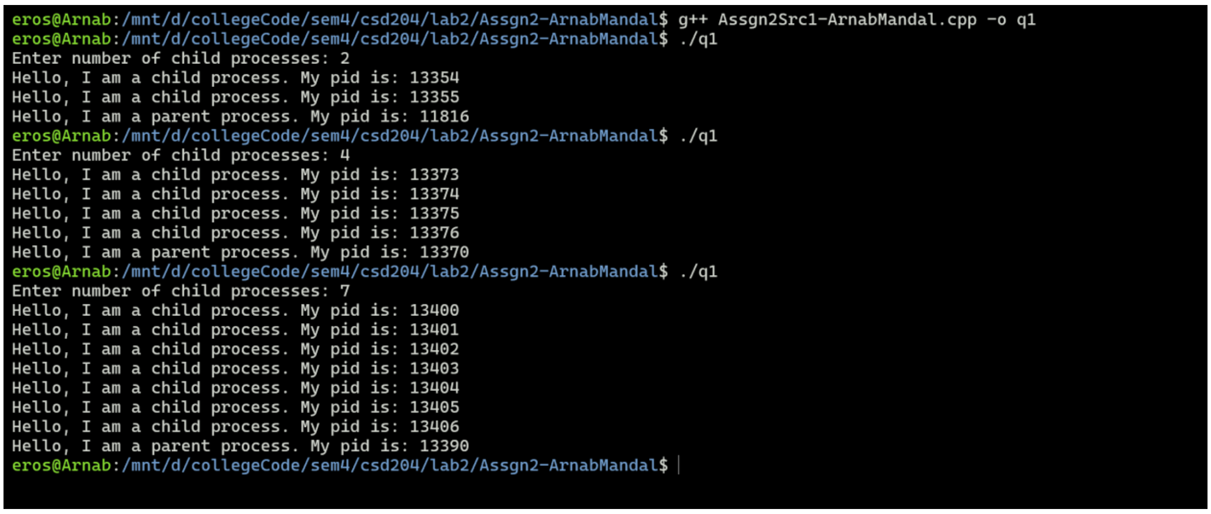
Q1.)

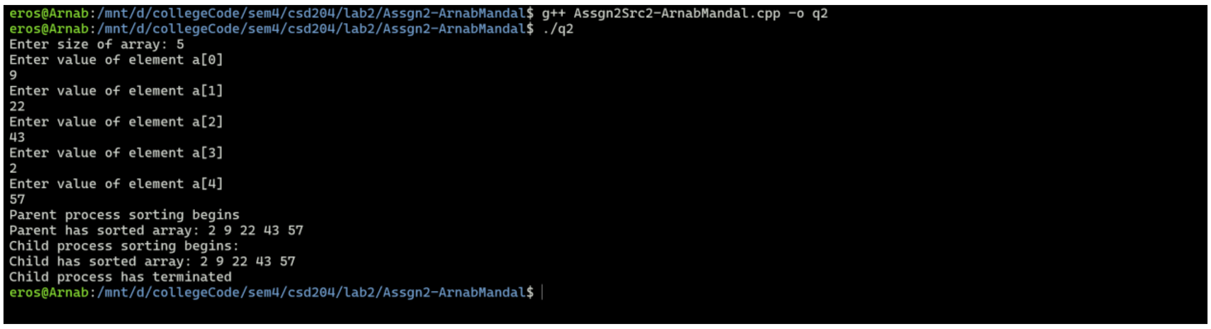


```
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab2/Assgn2-ArnabMandal$ g++ Assgn2Src1-ArnabMandal.cpp -o q1
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab2/Assgn2-ArnabMandal$ ./q1
Enter number of child processes: 2
Hello, I am a child process. My pid is: 13354
Hello, I am a child process. My pid is: 13355
Hello, I am a parent process. My pid is: 11816
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab2/Assgn2-ArnabMandal$ ./q1
Enter number of child processes: 4
Hello, I am a child process. My pid is: 13373
Hello, I am a child process. My pid is: 13374
Hello, I am a child process. My pid is: 13375
Hello, I am a child process. My pid is: 13376
Hello, I am a parent process. My pid is: 13370
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab2/Assgn2-ArnabMandal$ ./q1
Enter number of child processes: 7
Hello, I am a child process. My pid is: 13400
Hello, I am a child process. My pid is: 13401
Hello, I am a child process. My pid is: 13402
Hello, I am a child process. My pid is: 13403
Hello, I am a child process. My pid is: 13404
Hello, I am a child process. My pid is: 13405
Hello, I am a child process. My pid is: 13406
Hello, I am a parent process. My pid is: 13390
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab2/Assgn2-ArnabMandal$
```

We observe that, for input of N, we get N child processes, and N+1 total processes.

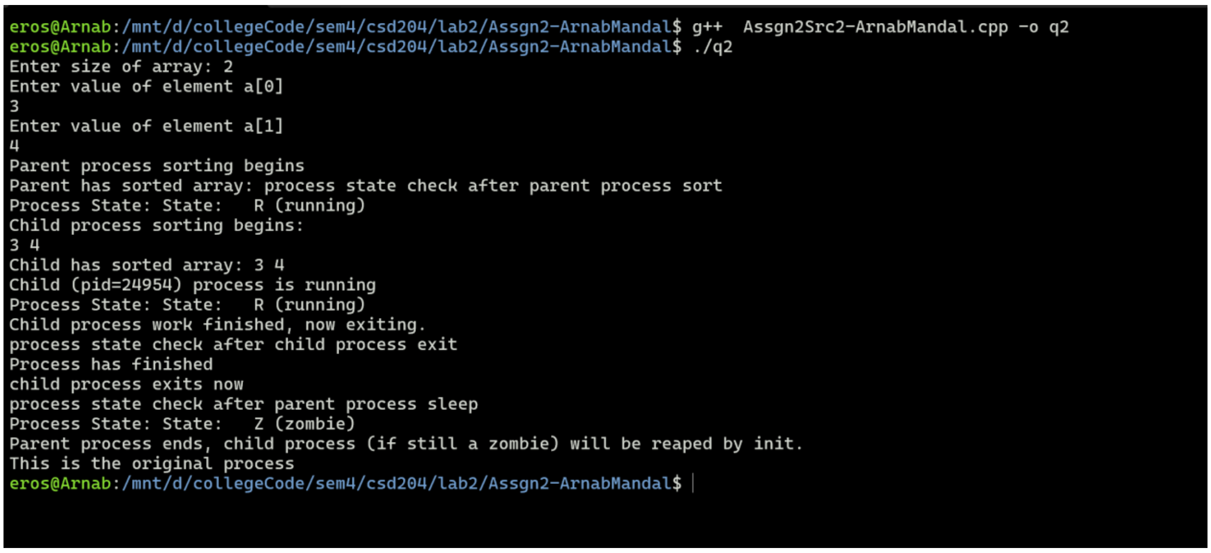
Q2.)

Sorting is carried out as normal and instructed in question.



```
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab2/Assgn2-ArnabMandal$ g++ Assgn2Src2-ArnabMandal.cpp -o q2
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab2/Assgn2-ArnabMandal$ ./q2
Enter size of array: 5
Enter value of element a[0]
9
Enter value of element a[1]
22
Enter value of element a[2]
43
Enter value of element a[3]
2
Enter value of element a[4]
57
Parent process sorting begins
Parent has sorted array: 2 9 22 43 57
Child process sorting begins:
Child has sorted array: 2 9 22 43 57
Child process has terminated
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab2/Assgn2-ArnabMandal$
```

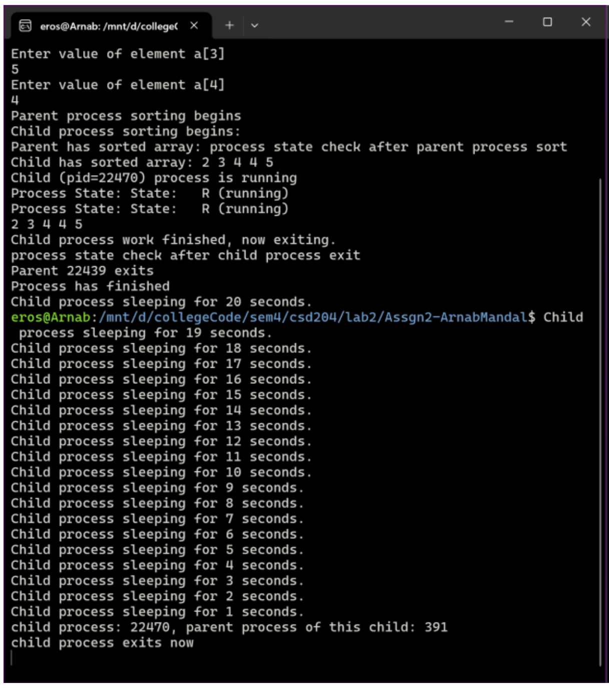
For observing, zombie process:
A zombie process occurs when its parent doesn't use WAIT() to collect the exit status of the child process. Here the child process becomes a zombie process until the parent collects its status update.



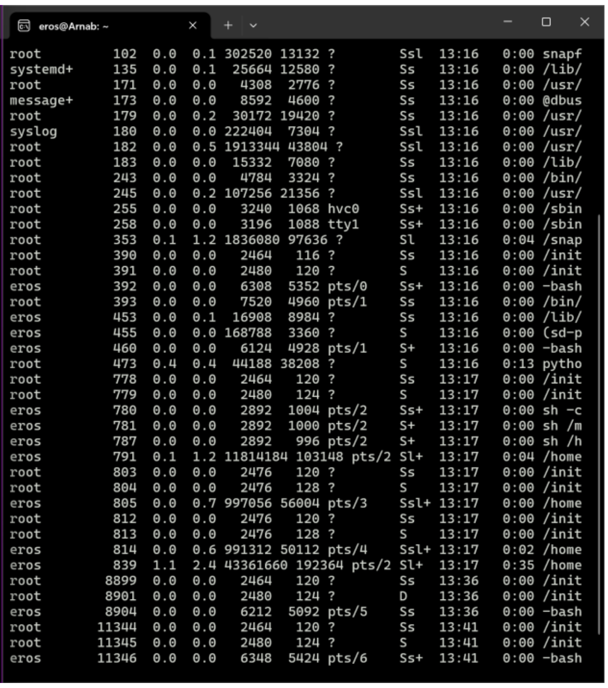
```
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab2/Assgn2-ArnabMandal$ g++  Assgn2Src2-ArnabMandal.cpp -o q2
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab2/Assgn2-ArnabMandal$ ./q2
Enter size of array: 2
Enter value of element a[0]
3
Enter value of element a[1]
4
Parent process sorting begins
Parent has sorted array: process state check after parent process sort
Process State: State:   R (running)
Child process sorting begins:
3 4
Child has sorted array: 3 4
Child (pid=24954) process is running
Process State: State:   R (running)
Child process work finished, now exiting.
process state check after child process exit
Process has finished
child process exits now
process state check after parent process sleep
Process State: State:   Z (zombie)
Parent process ends, child process (if still a zombie) will be reaped by init.
This is the original process
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab2/Assgn2-ArnabMandal$ |
```

For observing orphan process:

For a orphan process to occur, the parent must exit before the child process has finished running. In this case, it is adopted by init process and finishes running.



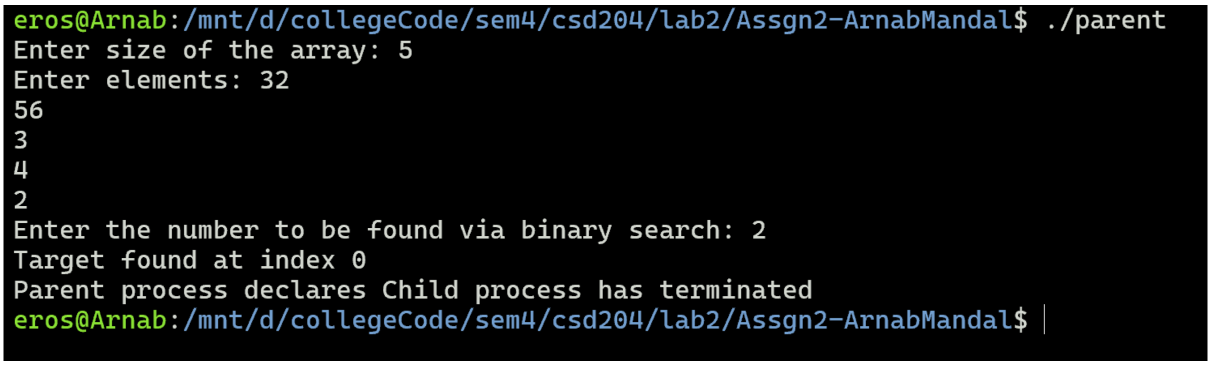
```
Enter value of element a[3]
5
Enter value of element a[4]
4
Parent process sorting begins
Child process sorting begins:
Parent has sorted array: process state check after parent process sort
Child has sorted array: 2 3 4 4 5
Child (pid=22470) process is running
Process State: State:   R (running)
Process State: State:   R (running)
2 3 4 4 5
Child process work finished, now exiting.
process state check after child process exit
Parent 22439 exits
Process has finished
Child process sleeping for 20 seconds.
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab2/Assgn2-ArnabMandal$ Child
 process sleeping for 19 seconds.
Child process sleeping for 18 seconds.
Child process sleeping for 17 seconds.
Child process sleeping for 16 seconds.
Child process sleeping for 15 seconds.
Child process sleeping for 14 seconds.
Child process sleeping for 13 seconds.
Child process sleeping for 12 seconds.
Child process sleeping for 11 seconds.
Child process sleeping for 10 seconds.
Child process sleeping for 9 seconds.
Child process sleeping for 8 seconds.
Child process sleeping for 7 seconds.
Child process sleeping for 6 seconds.
Child process sleeping for 5 seconds.
Child process sleeping for 4 seconds.
Child process sleeping for 3 seconds.
Child process sleeping for 2 seconds.
Child process sleeping for 1 seconds.
child process: 22470, parent process of this child: 391
child process exits now
|
```



```
root         102  0.0  0.1 302520 13132 ?        Ssl  13:16   0:00 snapf
systemd+     135  0.0  0.1  25664 12580 ?        Ss   13:16   0:00 /lib/
root         171  0.0  0.0   4308  2776 ?        Ss   13:16   0:00 /usr/
message+     173  0.0  0.0   8592  4600 ?        Ss   13:16   0:00 @dbus
root         179  0.0  0.2  30172 19420 ?        Ss   13:16   0:00 /usr/
syslog       180  0.0  0.0 222404  7304 ?        Ssl  13:16   0:00 /usr/
root         182  0.0  0.5 1913344 43804 ?       Ssl  13:16   0:00 /usr/
root         183  0.0  0.0  15332  7080 ?        Ss   13:16   0:00 /lib/
root         243  0.0  0.0   4784  3324 ?        Ss   13:16   0:00 /bin/
root         245  0.0  0.2 107256 21356 ?        Ssl  13:16   0:00 /usr/
root         255  0.0  0.0   3240  1068 hvc0     Ss+  13:16   0:00 /sbin
root         258  0.0  0.0   3196  1088 tty1     Ss+  13:16   0:00 /sbin
root         353  0.1  1.2 1836080 97636 ?       Sl   13:16   0:04 /snap
root         390  0.0  0.0   2464   116 ?        Ss   13:16   0:00 /init
root         391  0.0  0.0   2480   120 ?        S    13:16   0:00 /init
eros         392  0.0  0.0   6308  5352 pts/0    Ss+  13:16   0:00 -bash
root         393  0.0  0.0   7520  4960 pts/1    Ss   13:16   0:00 /bin/
eros         453  0.0  0.1  16908  8984 ?        Ss   13:16   0:00 /lib/
eros         455  0.0  0.0 168788  3360 ?        S    13:16   0:00 (sd-p
eros         460  0.0  0.0   6124  4928 pts/1    S+   13:16   0:00 -bash
root         473  0.4  0.4  44188 38208 ?        S    13:16   0:13 pytho
root         778  0.0  0.0   2464   120 ?        Ss   13:17   0:00 /init
root         779  0.0  0.0   2480   124 ?        S    13:17   0:00 /init
eros         780  0.0  0.0   2892  1004 pts/2    Ss+  13:17   0:00 sh -c
eros         781  0.0  0.0   2892  1000 pts/2    S+   13:17   0:00 sh /m
eros         787  0.0  0.0   2892   996 pts/2    S+   13:17   0:00 sh /h
eros         791  0.1  1.2 11814184 103148 pts/2 Sl+  13:17   0:04 /home
root         803  0.0  0.0   2476   120 ?        Ss   13:17   0:00 /init
root         804  0.0  0.0   2476   128 ?        S    13:17   0:00 /init
eros         805  0.0  0.7 997056 56004 pts/3    Ssl+ 13:17   0:00 /home
root         812  0.0  0.0   2476   120 ?        Ss   13:17   0:00 /init
root         813  0.0  0.0   2476   128 ?        S    13:17   0:00 /init
eros         814  0.0  0.6 991312 50112 pts/4    Ssl+ 13:17   0:02 /home
eros         839  1.1  2.4 43361660 192364 pts/2 Sl+  13:17   0:35 /home
root        8899  0.0  0.0   2464   120 ?        Ss   13:36   0:00 /init
root        8901  0.0  0.0   2480   124 ?        D    13:36   0:00 /init
eros        8904  0.0  0.0   6212  5092 pts/5    Ss   13:36   0:00 -bash
root       11344  0.0  0.0   2464   120 ?        Ss   13:41   0:00 /init
root       11345  0.0  0.0   2480   124 ?        S    13:41   0:00 /init
eros       11346  0.0  0.0   6348  5424 pts/6    Ss+  13:41   0:00 -bash
```

Here, the child process's parent (i.e. the init process) has pid:391. This is likely due to wsl's inbuilt feature of multiple forks of the init process, each being used for multiple different processes. Creating of orphan is regardless successful.

Q3.)



```
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab2/Assgn2-ArnabMandal$ ./parent
Enter size of the array: 5
Enter elements: 32
56
3
4
2
Enter the number to be found via binary search: 2
Target found at index 0
Parent process declares Child process has terminated
eros@Arnab:/mnt/d/collegeCode/sem4/csd204/lab2/Assgn2-ArnabMandal$ 
```

The parent first sorts the array into,

2, 3, 4, 32, 56 using bubble sort.

Then it locates 2 successfully at index 0 using binary search.