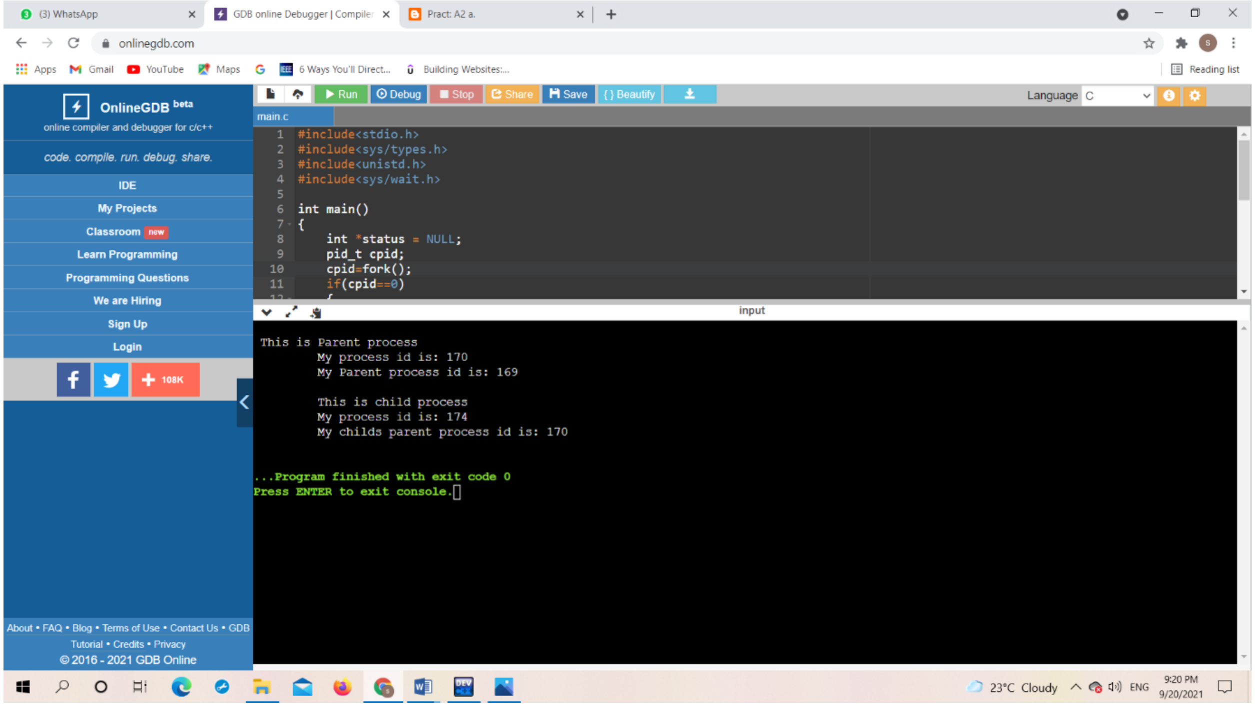
Title: Process control system calls: The demonstration of FORK and WAIT system calls along with zombie and orphan states.

Problem Statement: a. Implement the C program in which main program accepts the integers to be sorted. Main program uses the FORK system call to create a new process called a child process. Parent process sorts the integers using a sorting algorithm and waits for the child process using the WAIT system call to sort the integers using any sorting algorithm. Also demonstrate zombie and orphan states.

//For Zombie State:

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

int main()
{
   int *status = NULL;
   pid_t cpid;
   cpid = fork();
   if (cpid == 0)
   {
       sleep(5);
       printf("\n\t This is child process");
       printf("\n\t My process id is: %d", getpid());
       printf("\n\t My childs parent process id is: %d\n", getppid());
   }
   else
   {
       sleep(5);
       printf("\n This is Parent process");
       printf("\n\t My process id is: %d", getpid());
       printf("\n\t My Parent process id is: %d\n", getppid());
   }
   return 0;
}
```

OUTPUT:-

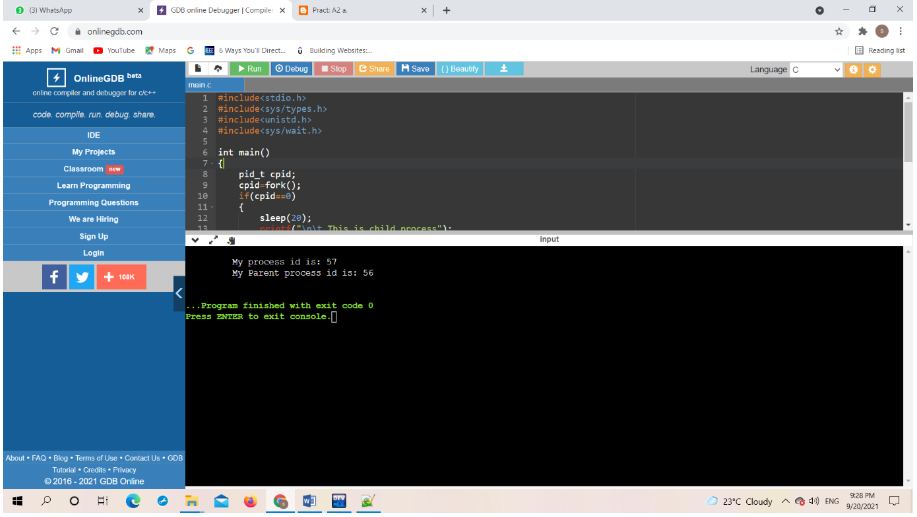


```c
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
#include<sys/wait.h>

int main()
{
    int *status = NULL;
    pid_t cpid;
    cpid=fork();
    if(cpid==0)
```

```
This is Parent process
        My process id is: 170
        My Parent process id is: 169

        This is child process
        My process id is: 174
        My childs parent process id is: 170


...Program finished with exit code 0
Press ENTER to exit console.
```

//For Orphan state

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

int main()
{
   pid_t cpid;
   cpid = fork();
   if (cpid == 0)
   {
       sleep(20);
       printf("\n\t This is child process");
       printf("\n\t My process id is: %d", getpid());
       printf("\n\t My Parent process id is: %d\n", getppid());
   }
   else
   {
       sleep(2);
       printf("\n\t My process id is: %d", getpid());
       printf("\n\t My Parent process id is: %d\n", getppid());
   }
   return 0;
}
```

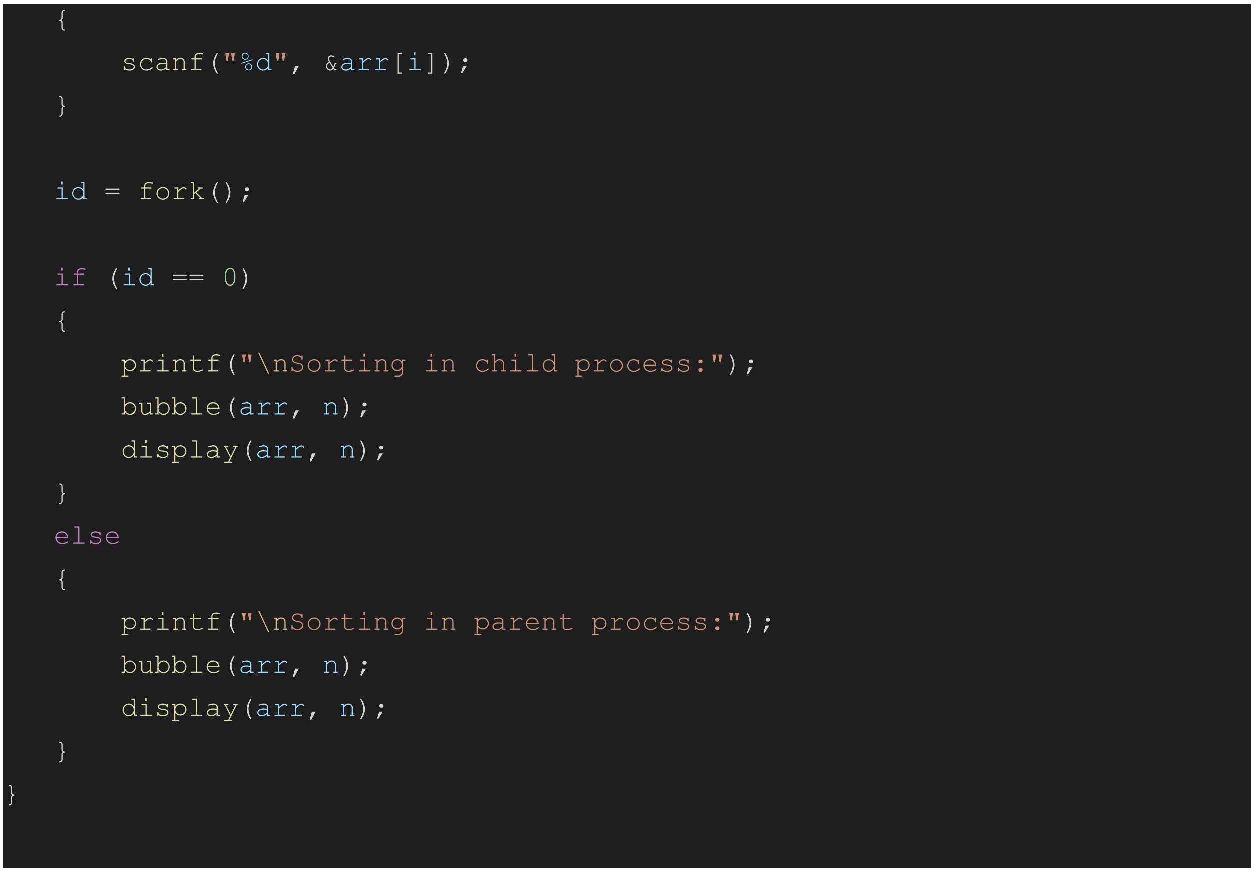


```c
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
#include<sys/wait.h>

int main()
{
    pid_t cpid;
    cpid=fork();
    if(cpid==0)
    {
        sleep(20);
        printf("\n\t This is child process");
```

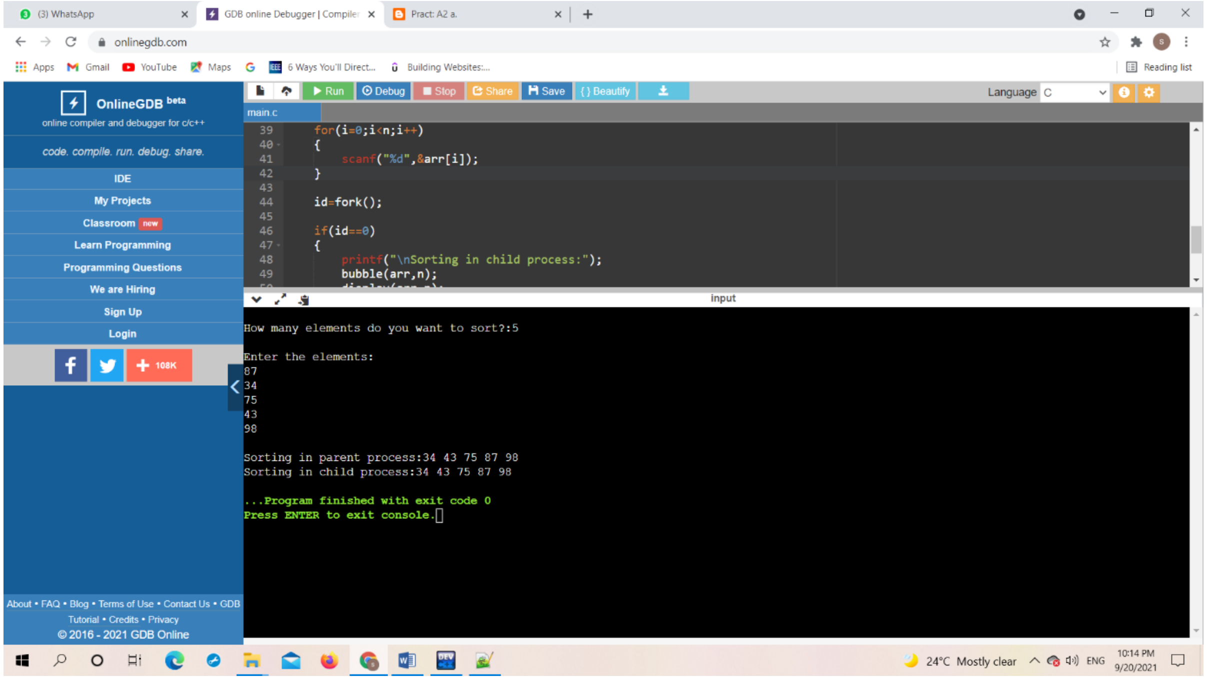
```
        My process id is: 57
        My Parent process id is: 56


...Program finished with exit code 0
Press ENTER to exit console.
```

//Bubble sort

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

void bubble(int arr[], int n)
{
   int i, j, k;
   for (i = 0; i < (n - 1); i++)
   {
       for (j = 0; j < (n - i - 1); j++)
       {
           if (arr[j] > arr[j + 1])
           {
               k = arr[j];
               arr[j] = arr[j + 1];
               arr[j + 1] = k;
           }
       }
   }
}

void display(int arr[], int n)
{
   for (int i = 0; i < n; i++)
   {
       printf("%d ", arr[i]);
   }
}

int main()
{
   pid_t id;
   int arr[20], n, i;
   printf("\nHow many elements do you want to sort?:");
   scanf("%d", &n);
   printf("\nEnter the elements:\n");

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



```
    {
        scanf("%d", &arr[i]);
    }

    id = fork();

    if (id == 0)
    {
        printf("\nSorting in child process:");
        bubble(arr, n);
        display(arr, n);
    }
    else
    {
        printf("\nSorting in parent process:");
        bubble(arr, n);
        display(arr, n);
    }
}
```



```
    for(i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }

    id=fork();

    if(id==0)
    {
        printf("\nSorting in child process:");
        bubble(arr,n);
```

```
How many elements do you want to sort?:5

Enter the elements:
87
34
75
43
98

Sorting in parent process:34 43 75 87 98
Sorting in child process:34 43 75 87 98

...Program finished with exit code 0
Press ENTER to exit console.
```