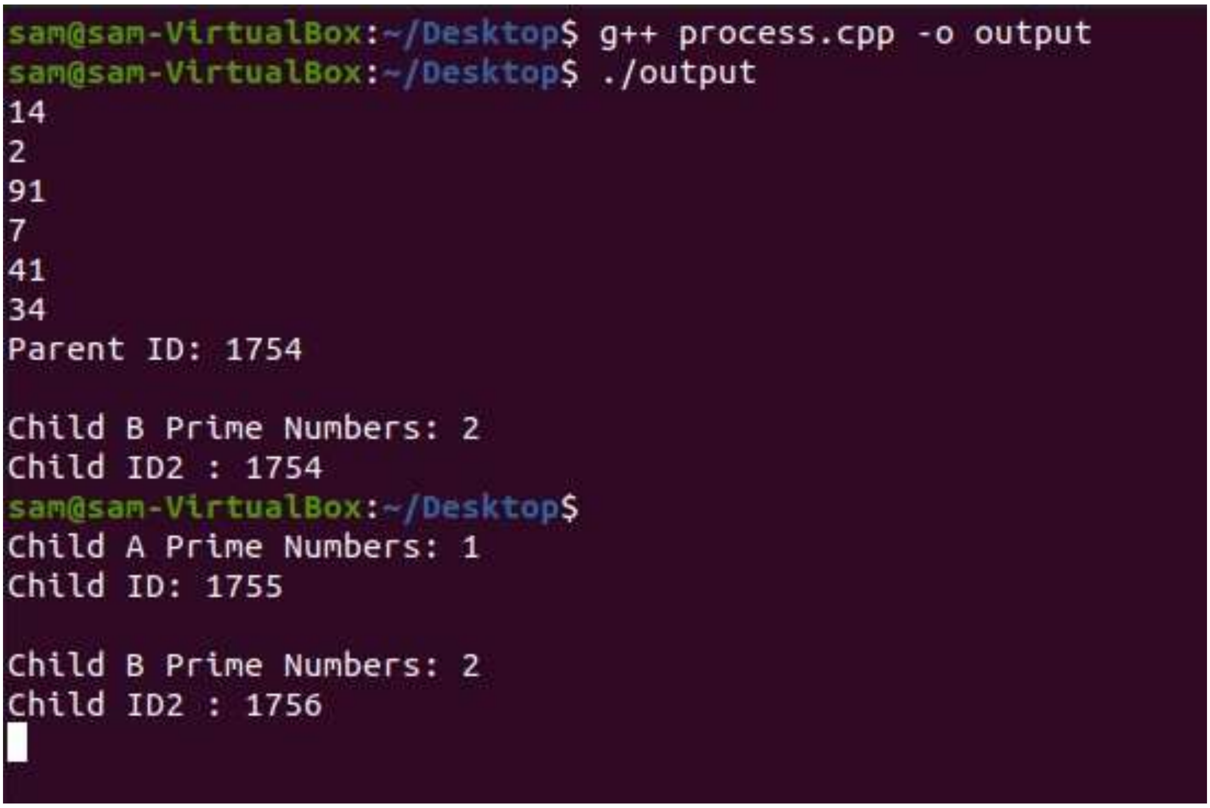
Using Fork, Exec, Wait & Exit System-Calls for Creating Chile Processes

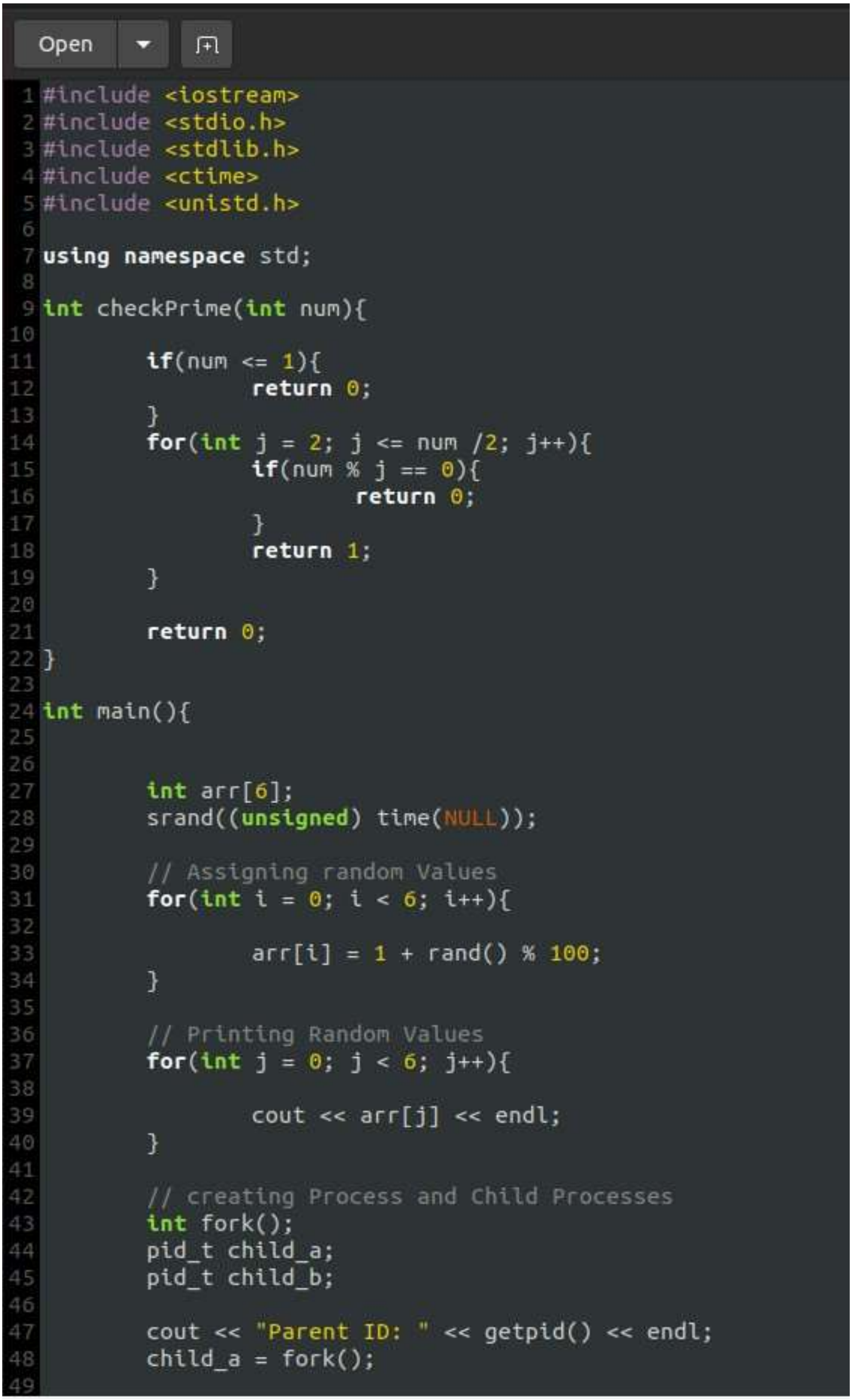
Task#1

Write a C++ program that creates an array of size 1000 and populates it with random integers between 1 and 100. Now, it creates two child processes. The first child process finds how many prime numbers are there among first 500 number while the second child process finds the number of prime numbers among the remaining 500 numbers.

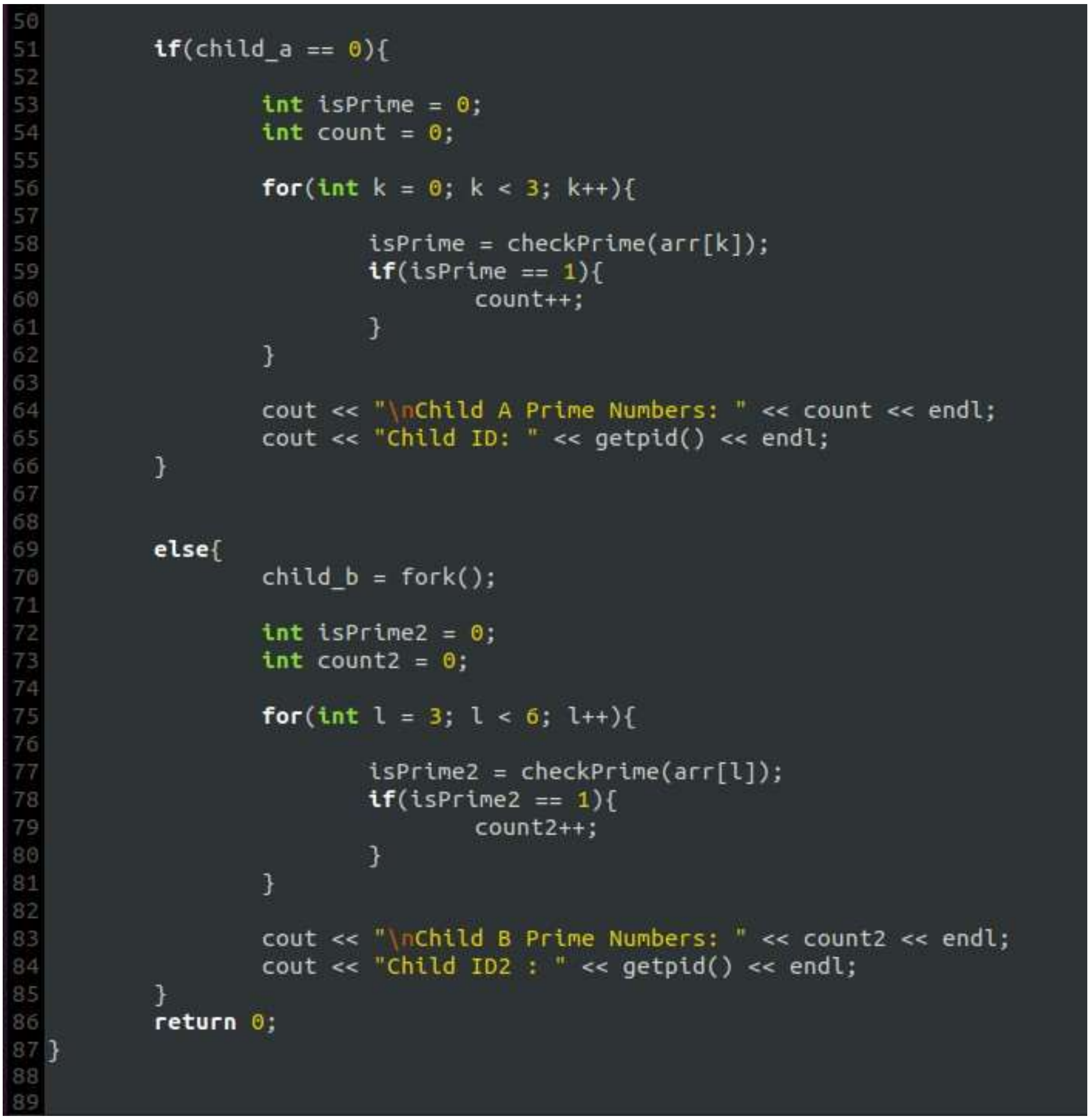
Solution: Question had 1000 digits but I checked it for only 6 just to display that the following program and processes work absolutely right. The program is tested for 1000 random values as well.



```
sam@sam-VirtualBox:~/Desktop$ g++ process.cpp -o output
sam@sam-VirtualBox:~/Desktop$ ./output
14
2
91
7
41
34
Parent ID: 1754

Child B Prime Numbers: 2
Child ID2 : 1754
sam@sam-VirtualBox:~/Desktop$
Child A Prime Numbers: 1
Child ID: 1755

Child B Prime Numbers: 2
Child ID2 : 1756
█
```



```cpp
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <ctime>
#include <unistd.h>

using namespace std;

int checkPrime(int num){

        if(num <= 1){
                return 0;
        }
        for(int j = 2; j <= num /2; j++){
                if(num % j == 0){
                        return 0;
                }
                return 1;
        }

        return 0;
}

int main(){


        int arr[6];
        srand((unsigned) time(NULL));

        // Assigning random Values
        for(int i = 0; i < 6; i++){

                arr[i] = 1 + rand() % 100;
        }

        // Printing Random Values
        for(int j = 0; j < 6; j++){

                cout << arr[j] << endl;
        }

        // creating Process and Child Processes
        int fork();
        pid_t child_a;
        pid_t child_b;

        cout << "Parent ID: " << getpid() << endl;
        child_a = fork();

```



```

        if(child_a == 0){

                int isPrime = 0;
                int count = 0;

                for(int k = 0; k < 3; k++){

                        isPrime = checkPrime(arr[k]);
                        if(isPrime == 1){
                                count++;
                        }
                }

                cout << "\nChild A Prime Numbers: " << count << endl;
                cout << "Child ID: " << getpid() << endl;
        }


        else{
                child_b = fork();

                int isPrime2 = 0;
                int count2 = 0;

                for(int l = 3; l < 6; l++){

                        isPrime2 = checkPrime(arr[l]);
                        if(isPrime2 == 1){
                                count2++;
                        }
                }

                cout << "\nChild B Prime Numbers: " << count2 << endl;
                cout << "Child ID2 : " << getpid() << endl;
        }
        return 0;
}


```