**Name: Swaraj Khan**

**Univ Roll no: 11500118032**

**5th sem C.S.E. Subject: OS Lab**

**Sub Code: PCC-CS592**

**Date: 23.02.2021**

**PCA -1**

**Problem 1:**

**Write a shell script to sort n numbers.**

**Code:**

**#Author : Swaraj**

**#Date : 23.02.2021**

**#Problem : Sort n numbers using shell script**

**echo "Enter number of elements :"**

**read n #number of elements in array**

**echo "Enter Numbers in array:"**

**for (( i = 0; i < $n; i++ )) #take elements of the array**

**do**

**read nos[$i]**

**done**

**for (( i = 0; i < $n ; i++ )) #applying bubble sort**

**do**

**for (( j = $i; j < $n; j++ ))**

**do**

**if [ ${nos[$i]} -gt ${nos[$j]} ]**

**then**

**t=${nos[$i]}**

**nos[$i]=${nos[$j]}**

**nos[$j]=$t**

**fi**

**done**

**done**

**echo -e "After applying Bubble Sort,the sorted numbers in ascending order : "**

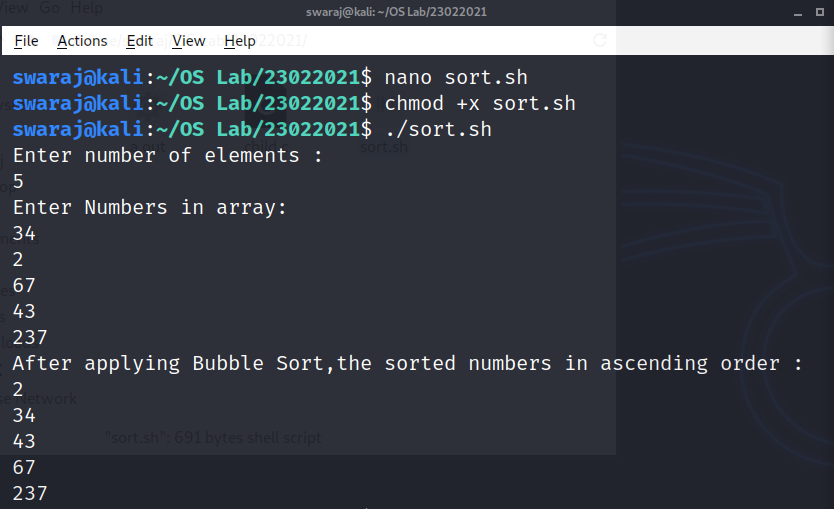
**for (( i=0; i < $n; i++ )) #print sorted numbers**

**do**

**echo ${nos[$i]}**

**done**

**Output:**



**Problem 2:**

**Write a program to create a child process and print the process related information**

**Code:**

**//Author: Swaraj Khan**

**//Date: 23.02.2021**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <unistd.h>**

**//n.b.: i have used sleep commands so that the fork() commands can be executed on the same process**

**int main()**

**{**

**int pid, pid1, pid2;**

**//variables to store the value returned from fork() system call**

**pid = fork();**

**// check whether process is parent or child**

**// if the pid is zero, then it is a child process**

**if (pid == 0) {**

**sleep(2);**

**printf("child[1] --> pid = %d and ppid = %d\n",**

**getpid(), getppid());**

**}**

**else {**

**pid1 = fork(); //creating second child process**

**if (pid1 == 0) {**

**sleep(1);**

**printf("child[2] --> pid = %d and ppid = %d\n",**

**getpid(), getppid());**

**}**

**else {**

**pid2 = fork(); //creating third child process**

**if (pid2 == 0) {**

**printf("child[3] --> pid = %d and ppid = %d\n",**

**getpid(), getppid());**

**}**

**//if the pid is greater than zero, then it is the parent process**

**else {**

**sleep(2);**

**printf("parent --> pid = %d\n", getpid());**

**}**

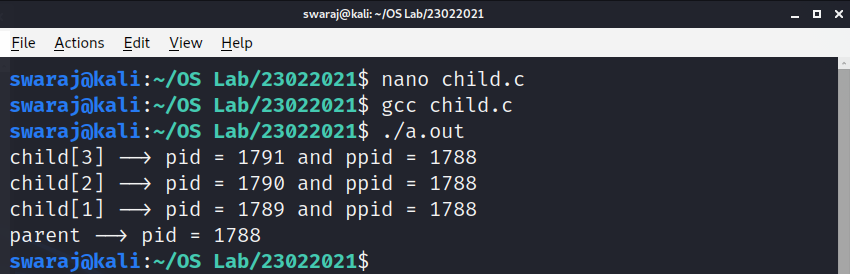
**}**

**}**

**return 0;**

**}**

**Output:**

****

**-----------------------**

**---------------**

**END**

**---------------**

**-----------------------**