Practical Ass 1

Set A

(1) Implement the C Program to create a child process using fork(), display parent and child process id. Child process will display the message “I am Child Process” and the parent process should display “I am Parent Process”.

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

int main()
{
   // fork() Create a child process

   int pid = fork();
   if (pid > 0) {
      printf("I am Parent process\n");
      printf("ID : %d\n\n", getpid());
   }
   else if (pid == 0) {
      printf("I am Child process\n");
      // getpid() will return process id of child process
      printf("ID: %d\n", getpid());

   }
   else {
      printf("Failed to create child process");
   }

   return 0;
}
```

```
I am Parent process
ID : 3698

I am Child process
ID: 3699
```

(2) Write a program that demonstrates the use of nice() system call. After a child process is started using fork(), assign higher priority to the child using nice() system call.

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
pid_t pid;
pid = fork();
if (pid == 0)
{
```

```
printf("\nI am child process, id=%d\n",getpid());
printf("\nPriority :%d,id=%d\n",nice (-7),getpid());
}
else
{
printf("\nI am parent process, id=%d\n",getpid());
nice(1);
printf("\nPriority :%d,id=%d\n",nice (15),getpid());
}
return 0;
}
```

```
I am parent process, id=6555
Priority :6,id=6555
I am child process, id=6556
Priority :-17,id=6556
```

Set B

(1) Implement the C program to accept n integers to be sorted. Main function creates child process using fork system call. Parent process sorts the integers using bubble sort and waits for child process using wait system call. Child process sorts the integers using insertion sort.

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

void bubblesort(int arr[30],int n)
{
        int i,j,temp;
        for(i=0;i<n;i++)
        {
                for(j=0;j<n-1;j++)
                {
                        if(arr[j]>arr[j+1])
                        {
                                temp=arr[j];
                                arr[j]=arr[j+1];
                                arr[j+1]=temp;
                        }
                }
        }
}

void insertionsort(int arr[30], int n)
{
```

```
    int i, j, temp;
    for (i = 1; i < n; i++) {
        temp = arr[i];
        j = i - 1;

        while(j>=0 && temp <= arr[j])
        {
            arr[j+1] = arr[j];
            j = j-1;
        }
        arr[j+1] = temp;
    }
}
void fork1()
{
            int arr[25],arr1[25],n,i,status;
            printf("\nEnter the no of values in array :");
            scanf("%d",&n);
            printf("\nEnter the array elements :");
            for(i=0;i<n;i++)
                        scanf("%d",&arr[i]);
            int pid=fork();
            if(pid==0)
            {
                        sleep(10);
                        printf("\nchild process\n");
                        printf("child process id=%d\n",getpid());
                        insertionsort(arr,n);
                        printf("\nElements Sorted Using insertionsort:");
                        printf("\n");
                        for(i=0;i<n;i++)
                                    printf("%d,",arr[i]);
                        printf("\b");
                        printf("\nparent process id=%d\n",getppid());
                        system("ps -x");
       }
       else
       {
                        printf("\nparent process\n");
                        printf("\nparent process id=%d\n",getppid());
                        bubblesort(arr,n);
                        printf("Elements Sorted Using bubblesort:");
                        printf("\n");
                        for(i=0;i<n;i++)
                                    printf("%d,",arr[i]);
                        printf("\n\n\n");
    }
 }
 int main()
```

```
{
        fork1();
        return 0;
}
```

```
Enter the no of values in array :5

Enter the array elements :2 3 4 1 6

parent process

parent process id=3610
Elements Sorted Using bubble sort:
1,2,3,4,6,
```

(2) Write a C program to illustrate the concept of orphan process. Parent process creates a child and terminates before child has finished its task. So child process becomes orphan process. (Use fork(), sleep(), getpid(), getppid()).

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
 int pid;
 pid=getpid();
 printf("Current Process ID is : %d\n",pid);
 printf("\n[Forking Child Process ... ] \n");
 pid=fork();
 if(pid < 0)
 {
  printf("\nProcess can not be created ");
 }
 else
 {
        if(pid==0)
  {
   printf("\nChild Process is Sleeping ...");
   sleep(5);
   printf("\nOrphan Child's Parent ID : %d",getppid());
  }

  else
  { /* Parent Process */
   printf("\nParent Process Completed ...");
  }
 }
 return 0;
```

```
}
Current Process ID is : 5546

[Forking Child Process ... ]

Parent Process Completed ...
Child Process is Sleeping ...
```