NAME : Bryce Ferreira

SUBJECT : SPOS LAB

ROLL NO.: 2203129

CLASS : TY-CORE-3

**Assignment 5**

# Aim –

Process control system calls - Fork, execve and wait system calls along with thedemonstrationof zombie and orphan states.

a) Application should consist of Fork-wait combination (parent with one application and child with another application) and students must demonstrate zombie and orphan states.b) Application should consist of Fork-execve combination (parent with one application and child with another application).

## Code :first file

#include<stdio.h>

#include <unistd.h>

#include<sys/types.h>

void sort(int a[10]);

int main(int argc,char \*argv[])

{ intpid; int i=0,n=10,search; char \*env[]={NULL}; int a[11];

char \*newarg[]={NULL,"sort.txt", NULL}; newarg[0]=argv[1];

printf("Enter array elements : "); for(i=1;i<=10;i++) scanf("%d",&a[i]);

printf("Enter value to find : "); scanf("%d", &search); FILE \*f; pid=fork(); if(pid==0) { sleep(1);

execve(argv[1],newarg,env);

} else { sort(a); f=fopen("sort.txt","w");

fprintf(f," %d",search); for(i=1;i<=n;i++) {

fprintf(f," %d",a[i]);

}

} fclose(f); }

void sort(int a[10])

{ int n=10, i=0, j=0, temp; for(i=1; i<=n; i++)

{

for(j=1; j<n; j++)

{ if(a[i]<a[j]) { temp=a[i]; a[i]=a[j]; a[j]=temp;

}

}

}

}

second file

#include<stdio.h>

#include<sys/types.h>

void bsearch(int a[10], int search);

int main(int argc,char \*argv[])

{

int a[11],i,n=10,search,first, last, middle,flag=0;

FILE \*f;

f=fopen(argv[1],"r");

printf(" %s",argv[1]); fscanf(f,"%d",&search);

printf(" Key=%d\n",search);

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

{

fscanf(f,"%d",&a[i]);

printf(" %d",a[i]);

} first=0; last=n-1; middle=(first+last)/2; while(first<=last)

{

if(a[middle]<search){

first= middle+1; middle=(first+last)/2;

}

else if(a[middle]==search)

{

printf("\n%dElement found at location %d \n", search, middle+1); flag=1; break; } else { last=middle-1; middle=(first+last)/2;

}

} if(flag==0)

printf("\n Not found");

return(0);

}

void bsearch(int a[11], int search)

{

int i, first, last, middle, n=10;

first=0; last=n-1;

middle=(first+last)/2;

while(first<=last)

{ if(a[middle]<search)

first= middle+1;

else if(a[middle]==search)

{

printf("%d Elementm found at location %d \n", search, middle+1); break; }

else { last=middle-1; middle=(first+last)/2;

}

if(first>last)

{

printf("Element not found %d is not present in the list\n", search);

//return 0;

}

}

}

**Output**

