**PART A**

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

void quickSort(int [],int ,int );

int partition(int [],int ,int );

void mergeSort(int [],int ,int );

void merge(int [],int ,int ,int ,int );

int main()

{

int i,j,n;

int \*status=NULL;

int arr[30];

printf("\nEnter the number of elements:");

scanf("%d",&n);

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

{

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

}

pid\_t pid;

pid=fork();

if(pid==0)

{

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", getppid());

quickSort(arr,0,n-1);

printf("\nQuicksort");

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

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

printf("\n\n");

}

else

{

printf("\n\n\t Parent process resumed after the execution of child process with PID %d", pid);

printf("\n\t My process id is : %d", getpid());

printf("\n\t My Parent process id is : %d", getppid());

mergeSort(arr,0,n-1);

printf("\nMergesort:");

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

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

printf("\n\n");

pid=wait(status);

}

}

void quickSort(int arr[],int low,int high)

{

int j;

if(low<high)

{

j=partition(arr,low,high);

quickSort(arr,low,j-1);

quickSort(arr,j+1,high);

}

}

int partition(int arr[],int low,int high)

{

int i,j,temp,pivot;

pivot=arr[low];

i=low;

j=high+1;

do

{

do

i++;

while(arr[i]<pivot && i<=high);

do

j--;

while(arr[j]>pivot);

if(i<j)

{

temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

while(i<j);

arr[low]=arr[j];

arr[j]=pivot;

return(j);

}

void mergeSort(int arr[],int low,int high)

{

int mid;

if(low<high)

{

mid=(low+high)/2;

mergeSort(arr,low,mid);

mergeSort(arr,mid+1,high);

merge(arr,low,mid,mid+1,high);

}

}

void merge(int arr[],int i1,int j1,int i2,int j2)

{

int temp[50];

int i,j,k;

i=i1;

j=i2;

k=0;

while(i<=j1 && j<=j2)

{

if(arr[i]<arr[j])

temp[k++]=arr[i++];

else

temp[k++]=arr[j++];

}

while(i<=j1)

temp[k++]=arr[i++];

while(j<=j2)

temp[k++]=arr[j++];

for(i=i1,j=0;i<=j2;i++,j++)

arr[i]=temp[j];

}

**PART B**

**Main.c**

#include <sys/types.h>

#include <sys/wait.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

void quicksort(int [10],int,int);

void main()

{

int pid, ppid, a[10], size, i;

char \*argv[12];

for(i=0; i<12; i++)

{

argv[i] = (char \*) malloc(20);

}

printf("\nIn the main of parent process. \nMy process-id is %d.", getpid());

printf("\n\nEnter the number of elements in the array: \n");

scanf("%d", &size);

printf("Enter %d elements: ", size);

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

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

quicksort(a, 0, size-1);

printf("Sorted elements: ");

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

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

printf("\nPARENT: I am calling my child for searching now!\n");

printf("-------------------------------------------------------");

pid = vfork(); //Here, the child process is created and both

//simultaneously. fork() returns 0 to child process and pid of

//child to parent process. So, if int

//pid here is 0, it means we

//are in child process and if any

//+ve no., then in parent process

//where the +ve no. is the child's pid

if(pid==0)

{

printf("CHILD: In child process. My process-id is %d.\n", getpid());

printf("CHILD: My parent's process-id is %d.\n\n", getppid());

printf("\n CHILD: Calling searching using execv now. Will be overlayed!\n");

for(i=0; i<size; i++){

sprintf(argv[i+1], "%d", a[i]);

}

argv[0] = "./child";

argv[size+2] = NULL;

execv("./child", argv);

printf("Error");

printf("\nCHILD: I am dying now\n");

printf("-------------------------------------------------------\n\n");

}

else if (pid>0)

{

system("wait");

printf("\nPARENT: I am dying now\n");

printf("-------------------------------------------------------");

printf("\n\n.");

}

}

void quicksort(int a[10],int first,int last)

{

int pivot,j,temp,i;

if(first<last)

{

pivot=first;

i=first;

j=last;

while(i<j){

while(a[i]<=a[pivot]&&i<last)

i++; while(a[j]>a[pivot])

j--;

if(i<j){

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

temp=a[pivot];

a[pivot]=a[j];

a[j]=temp;

quicksort(a,first,j-1);

quicksort(a,j+1,last);

}

}

Child.c

#include <sys/types.h>

#include <sys/wait.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

void binarysearch(long int [10], int n);

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

{

int i, n=0;

long ret[10];

for(i=1; argv[i]!=NULL; i++)

{

ret[i] = strtol(argv[i], NULL, 10);

n++;

}

printf("SEARCH: In second child process. My process-id is %d.\n", getpid());

printf("SEARCH: My parent's process-id is %d.\n\n", getppid());

binarysearch(ret, n);

}

void binarysearch(long int a[10], int n)

{

int c, first, last, middle, search;

printf("Enter value to search: \n");

scanf("%d",&search);

printf("\n\n=================In Binary Search Program===================\n");

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 found at location %d.\n", search, middle);

break;

}

else

last = middle - 1;

middle = (first + last)/2;

}

if ( first > last )

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

}