

Assignment 2-A

code:

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

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

void mergeSort(int arr[],int low,int mid,int high)
{
    int i,j,k,l,b[20];
    l=low;
    i=low;
    j=mid+1;

    while((l<=mid)&&(j<=high)){
        if(arr[l]<=arr[j]){
            b[i]=arr[l];
            l++;
        }
        else{
            b[i]=arr[j];
            j++;
        }

        i++;
    }

    if(l>mid){
        for(k=j;k<=high;k++){
            b[i]=arr[k];
            i++;
        }
    }
    else{
        for(k=l;k<=mid;k++){
            b[i]=arr[k];
            i++;
        }
    }

    for(k=low;k<=high;k++)
    {
```

```

    arr[k]=b[k];
}
}

```

```

void partition(int arr[],int low,int high)
{
    int mid;
    if(low<high)
    {
        double temp;

        mid=(low+high)/2;
        partition(arr,low,mid);
        partition(arr,mid+1,high);
        mergeSort(arr,low,mid,high);
    }
}

```

```

void display(int a[],int size){
    int i;
    for(i=0;i<size;i++){
        printf("%d\t\t",a[i]);
    }
    printf("\n");
}

```

```

int main()
{
    int pid, child_pid;
    int size,i,status;

    /*   Input the Integers to be sorted   */
    printf("Enter the number of Integers to Sort:::\t");
    scanf("%d",&size);

    int a[size];
    int pArr[size];
    int cArr[size];

    for(i=0;i<size;i++){
        printf("Enter number %d:",(i+1));
        scanf("%d",&a[i]);
        pArr[i]=a[i];
        cArr[i]=a[i];
    }
}

```

```

/* Display the Entered Integers */

printf("Your Entered Integers for Sorting\n");
display(a,size);

/* Process ID of the Parent */

pid=getpid();
printf("Current Process ID is : %d\n",pid);

/* Child Process Creation */

printf("[ Forking Child Process ... ] \n");
child_pid=fork(); /* This will Create Child Process and
                  Returns Child's PID */
if( child_pid < 0){

    /* Process Creation Failed ... */

    printf("\nChild Process Creation Failed!!!!\n");
    exit(-1);
}
else if( child_pid==0) {
/* Child Process */
printf("\nThe Child Process\n");
printf("\nchild process is %d",getpid());
printf("\nparent of child process is %d",getppid());
printf("Child is sorting the list of Integers by QUICK SORT::\n");
quickSort(cArr,0,size-1);
printf("The sorted List by Child::\n");
display(cArr,size);
printf("Child Process Completed ...\n");
sleep(10);
printf("\nparent of child process is %d",getppid());
}

else {
/* Parent Process */
printf("parent process %d started\n",getpid());
printf("Parent of parent is %d\n",getppid());

sleep(30);
printf("The Parent Process\n");
printf("Parent %d is sorting the list of Integers by MERGE SORT\n",pid);
partition(pArr,0,size-1);
printf("The sorted List by Parent::\n");

```

```

        display(pArr,size);
        // wait(&status);
        printf("Parent Process Completed ...\n");
    }

    return 0;
}

```

```

int split ( int a[ ], int lower, int upper )
{
    int i, p, q, t ;

    p = lower + 1 ;
    q = upper ;
    i = a[lower] ;

    while ( q >= p )
    {
        while ( a[p] < i )
            p++ ;

        while ( a[q] > i )
            q-- ;

        if ( q > p )
        {
            t = a[p] ;
            a[p] = a[q] ;
            a[q] = t ;
        }
    }

    t = a[lower] ;
    a[lower] = a[q] ;
    a[q] = t ;

    return q ;
}

```

```

void quickSort(int a[],int lower, int upper){
    int i ;
    if ( upper > lower )
    {
        i = split ( a, lower, upper ) ;
        quickSort ( a, lower, i - 1 ) ;
        quickSort ( a, i + 1, upper ) ;
    }
}

```

Output:

```
ak-linux-computer@fedora-ak:~ — ./test2
[ak-linux-computer@ak ~]$ gcc test2.c -o test2
[ak-linux-computer@ak ~]$ ./test2
Enter the number of Integers to Sort:::      5
Enter number 1:45
Enter number 2:87
Enter number 3:12
Enter number 4:72
Enter number 5:90
Your Entered Integers for Sorting
45          87          12          72          90
Current Process ID is : 128144
[ Forking Child Process ... ]
parent process 128144 started
Parent of parent is 128055

The Child Process

child process is 128152
parent of child process is 128144Child is sorting the list of Integers by QUICK SORT::
The sorted List by Child::
12          45          72          87          90
Child Process Completed ...

parent of child process is 128144
```

Assignment 2-B:

code:

//for child:

```
#include<stdio.h>
```

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

```
#include<unistd.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%d Element 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 Element 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;
}  
}  
}
```

```
//for main:
```

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

```
void sort(int a[10]);
```

```
int main(int argc,char *argv[])  
{  
int pid;  
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;
```

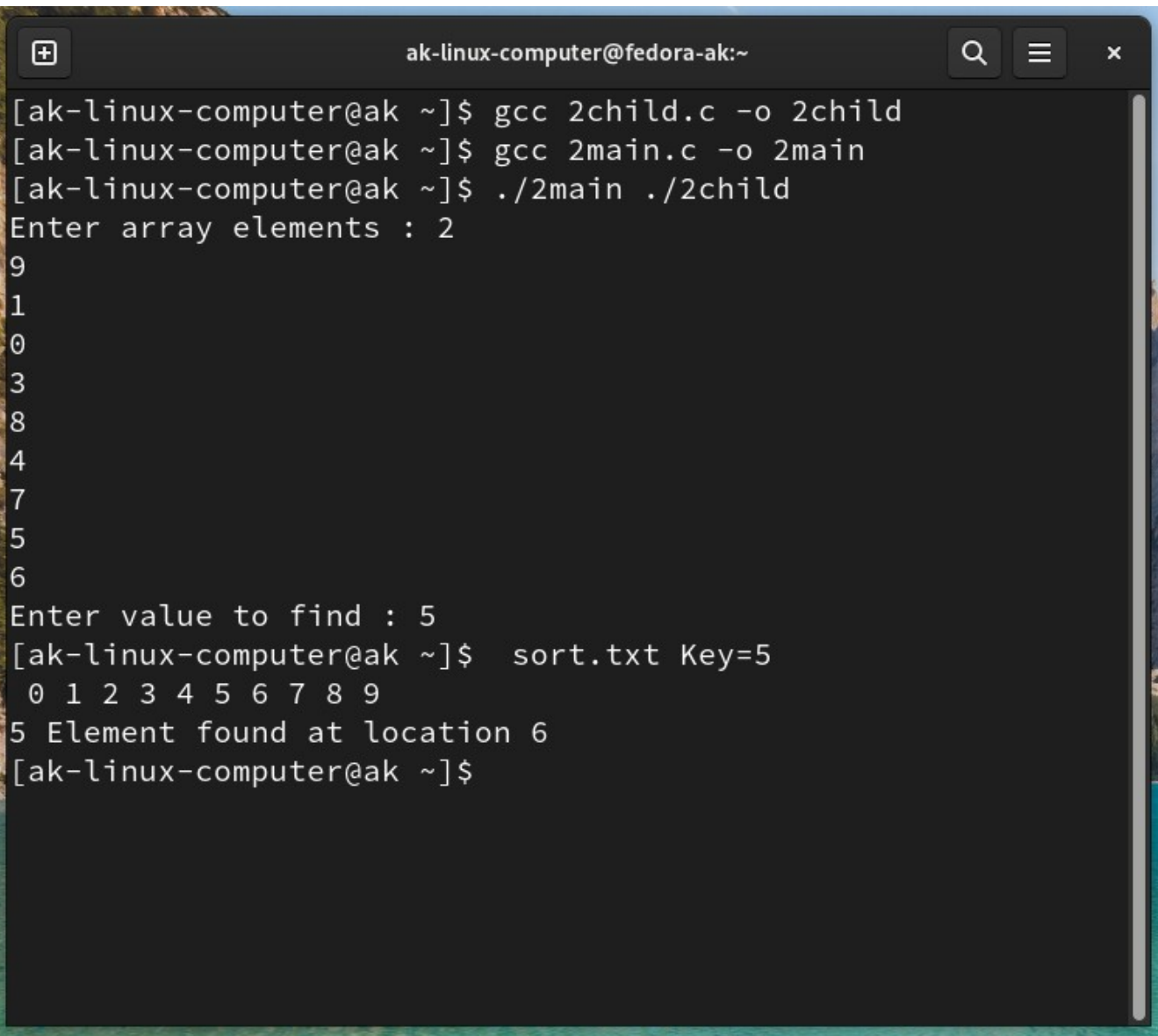
```
}
```

```
}
```

```
}
```

```
}
```

Output:

A terminal window titled 'ak-linux-computer@fedora-ak:~' with search, menu, and close icons in the title bar. The terminal shows the compilation and execution of a C program. The user enters '2' for the number of array elements, followed by the values 9, 1, 0, 3, 8, 4, 7, 5, and 6. Then, the user enters '5' as the value to find. The program outputs the sorted array indices and the location of the value 5.

```
[ak-linux-computer@ak ~]$ gcc 2child.c -o 2child
[ak-linux-computer@ak ~]$ gcc 2main.c -o 2main
[ak-linux-computer@ak ~]$ ./2main ./2child
Enter array elements : 2
9
1
0
3
8
4
7
5
6
Enter value to find : 5
[ak-linux-computer@ak ~]$ sort.txt Key=5
0 1 2 3 4 5 6 7 8 9
5 Element found at location 6
[ak-linux-computer@ak ~]$
```