**CODE:**

**Main Code :**

#include<stdio.h>

#include<stdlib.h>

#include<sys/wait.h>

#include<sys/types.h>

#include<unistd.h>

void Func(char\* arr[], int argc, char\* argv[])

{

int i;

for (i=1;i<argc;i++)

{

arr[i-1]=argv[i];

}

arr[i]= NULL;

printf("Array elements are: \n");

for (i=1;i<argc-1;i++)

{

printf("%s\t",arr[i]);

}

printf("\n");

int n= argc-2;

printf("Total Size of N is %d \n",n);

for (int j=1;j<argc-1;j++)

{

for (int k=1;k<argc-j-1;k++)

{

if(atoi(arr[k+1])<atoi(arr[k]))

{

char \*temp =arr[k];

arr[k]=arr[k+1];

arr[k+1]=temp;

}

}

}

printf("Array elements after sorting are: \n");

for (i=1;i<argc-1;i++)

{

printf("%s\t",arr[i]);

}

printf("\n");

}

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

{

//const char \*arr[]={"./b.out","10","20","30","40","50",NULL};

char \*arr[argc];

arr[0] = argv[0];

Func(arr,argc,argv);

pid\_t pid;

pid = fork();

if(pid == 0)

{

/\*

for (int i = 1; i < argc; i++)

{

arr[i] = argv[i];

}

arr[argc-1] = NULL; // Set the last element to NULL

\*/

printf("\nChild Process ID is %d ", getpid());

printf("\nChild's Parent Process ID is %d \n", getppid());

execv(arr[0], arr);

}

else

{

wait(NULL);

printf("\nParent Process ID is %d \n",getpid());

printf("Parent Parent's Process ID is %d \n",getppid());

}

}

/\*

gcc main.c -o main.out

gcc b.c -o b.out

gcc main.c

./a.out ./b.out 32 4 21 42 .....

\*/

**Source Code :**

#include <stdio.h>

#include <stdlib.h>

// Array Addition

void ArrayAddition(char\* arr[], int len)

{

int sum = 0;

for (int i = 1; i < len; i++)

{

sum += atoi(arr[i]);

}

printf("\nThe sum of %d Number's is -> %d \n", len - 1, sum);

}

// Storing Array

void ArrayStore(char\* argv[], int arr[], int len)

{

for (int i = 0; i < len; i++)

{

arr[i] = atoi(argv[i + 1]);

}

}

// Print Array

void PrintArray(int arr[], int len)

{

for (int i = 0; i < len; i++)

{

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

}

printf("\n");

}

// Sorting Values

void BubbleSort(int arr[], int len)

{

int temp;

for (int i = 0; i < len - 1; i++)

{

for (int j = 0; j < len - i - 1; j++)

{

if (arr[j] < arr[j + 1])

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

// Binary Search

void BinarySearch(int arr[], int low, int high, int findElement)

{

int mid;

while (low <= high)

{

mid = (low + high) / 2;

if (arr[mid] == findElement)

{

printf("\n%d Element is present at Index %d", arr[mid], mid);

break;

}

else if (arr[mid] > findElement)

{

low = mid + 1;

}

else if (arr[mid] < findElement)

{

high = mid - 1;

}

}

}

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

{

int arr[argc - 1];

ArrayAddition(argv, argc);

int len = argc - 1;

printf("\nStoring the array Element\n");

ArrayStore(argv, arr, len);

PrintArray(arr, len);

printf("\nPrinting Array Elements in Reverse Order\n");

BubbleSort(arr, len);

PrintArray(arr, len);

printf("\nEnter the Element You Want to Find -> ");

int findElement;

scanf("%d", &findElement);

BinarySearch(arr, 0, len - 1, findElement);

return 0;

}

**Output :**

┌──(pranav㉿Pranav)-[~/OS]

└─$ gcc b.c -o b.out

┌──(pranav㉿Pranav)-[~/OS]

└─$ gcc main.c

┌──(pranav㉿Pranav)-[~/OS]

└─$ ./a.out ./b.out 4 34 54 2 0 54 23 54 -34 53 -2 -5434

Array elements are:

4 34 54 2 0 54 23 54 -34 53 -2 -5434

Total Size of N is 12

Array elements after sorting are:

-5434 -34 -2 0 2 4 23 34 53 54 54 54

Child Process ID is 8940

Child's Parent Process ID is 8939

The sum of 12 Number's is -> -5192

Storing the array Element

-5434 -34 -2 0 2 4 23 34 53 54 54 54

Printing Array Elements in Reverse Order

54 54 54 53 34 23 4 2 0 -2 -34 -5434

Enter the Element You Want to Find -> -2

-2 Element is present at Index 9

Parent Process ID is 8939

Parent Parent's Process ID is 6112