Assignment – 2(a)

Roll no: 33229

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


void merge(int arr[], int l, int m, int r){
   int i, j, k;
   int n1 = m - l + 1;     int n2 = r - m;
   // Create temp arrays     int L[n1], R[n2];
   // Copy data to temp arrays L[] and R[]
   for (i = 0; i < n1; i++)        L[i] = arr[l + i];     for (j = 0; j < n2; j++)
      R[j] = arr[m + 1 + j];
   // Merge the temp arrays back into arr[l..r
   i = 0;     j = 0;     k = l;
   while (i < n1 && j < n2) {
      if (L[i] <= R[j]) {           arr[k] = L[i];           i++;
      }
      else {           arr[k] = R[j];           j++;
      }
      k++;
   }
   while (i < n1) {
      arr[k] = L[i];        i++;        k++;
   }
   while (j < n2) {        arr[k] = R[j];        j++;        k++;
   }
}

void mergeSort(int arr[], int l, int r)
{    if (l < r) {        int m = l + (r - l) / 2;
      // Sort first and second halves        mergeSort(arr, l, m);        mergeSort(arr, m +
1, r);        merge(arr, l, m, r);
   }
}

void printArray(int A[], int size){
   int i;
   for (i = 0; i < size; i++)        printf("%d ", A[i]);     printf("\n");
}
```

```
int main(){
	pid_t pid;	int n;

	printf("Enter the number of integers: ");		scanf("%d", &n);	int
arr[n];		printf("Enter the integers: ");		for (int i = 0; i < n; i++) {
	scanf("%d", &arr[i]);
	}
	printf("Given array is : ");		printArray(arr, n);

	printf("fork program starting\n");
	pid = fork();  switch(pid){
		case -1:
		perror("fork failed");
		exit(1);

		case 0:
		printf("This is the child\n");

		printf("The ppid of child current process : %d",getppid());
		printf("\n");
		printf("The pid of clild current process : %d",getpid());

		//orphan process
	sleep(5);
		system("ps -elf | grep a.out");		printf("Child process
exiting...\n");
		exit(0);
		break;

		default:
		printf("This is the parent Process Sorting of Array\n");
	printf("The ppid of parent current process : %d",getppid());
		printf("\n");
		printf("The pid of parent current process : %d",getpid());

		mergeSort(arr, 0, n - 1);

		printf("\nSorted array is \n");		printArray(arr, n);

		//zombie process
	sleep(10);		system("ps -elf | grep a.out");	wait(0);
	printf("Parent process exiting...\n");
		break;
	}
	exit(0);
```

}

OUTPUT:

```
Enter the number of integers: 5
Enter the integers: 4
2
1
7
3
Given array is : 4 2 1 7 3  fork program starting
This is the parent Process Sorting of Array
The ppid of parent current process : 3482
The pid of parent current process : 3511
Sorted array is
1 2 3 4 7
This is the child
The ppid of child current process : 3511
S soham    3511   3482  0  80   0 -   694 hrtime 23:33 pts/0    00:00:00 ./a.out
S soham    3559   3511  0  80   0 -   694 do_wai 23:34 pts/0    00:00:00 ./a.out
0 S soham    3560   3559  0  80   0 -   723 do_wai 23:34 pts/0    00:00:00 sh -c ps -elf | grep
a.out
0 S soham    3562   3560  0  80   0 -  4434 pipe_r 23:34 pts/0    00:00:00 grep a.out The
pid of clild current process : 3559Child process exiting...
S soham    3511   3482  0  80   0 -   694 do_wai 23:33 pts/0    00:00:00 ./a.out
Z soham    3559   3511  0  80   0 -     0 -      23:34 pts/0    00:00:00 [a.out] <defunct>
0 S soham    3563   3511  0  80   0 -   723 do_wai 23:34 pts/0    00:00:00 sh -c ps -elf | grep
a.out
0 S soham    3565   3563  0  80   0 -  4434 pipe_r 23:34 pts/0    00:00:00 grep a.out
Parent process exiting...
```

Assignmrnt 2(b)

Main.c

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


void merge(int arr[], int l, int m, int r){
```

```
   int i, j, k;
   int n1 = m - l + 1;     int n2 = r - m;
// Create temp arrays int L[n1], R[n2];
   // Copy data to temp arrays L[] and R[]
   for (i = 0; i < n1; i++)        L[i] = arr[l + i];     for (j = 0; j < n2; j++)
      R[j] = arr[m + 1 + j];
   // Merge the temp arrays back into arr[l..r
   i = 0;     j = 0;     k = l;
   while (i < n1 && j < n2) {
      if (L[i] <= R[j]) {           arr[k] = L[i];           i++;        }
      else {           arr[k] = R[j];           j++;
      }
      k++;
   }
   while (i < n1) {
      arr[k] = L[i];        i++;        k++;
   }
while (j < n2) {
      arr[k] = R[j];        j++;        k++;
   }
}

void mergeSort(int arr[], int l, int r)
{    if (l < r) {        int m = l + (r - l) / 2;
      // Sort first and second halves        mergeSort(arr, l, m);        mergeSort(arr, m + 1, r);
merge(arr, l, m, r);
   }
}
void printArray(int A[], int size)
{    int i;
   for (i = 0; i < size; i++)        printf("%d ", A[i]);     printf("\n");
}
```

```
int main() {

 int n, i;  pid_t pid;

        printf("Enter the number of elements: ");

        scanf("%d", &n);

        int arr[n];       printf("Enter the elements:\n");

        for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
        }
                mergeSort(arr, 0, n - 1);
                printf("\nSorted array is \n");
                printArray(arr, n);
                pid = fork();                s
                witch (pid) {
        case -1:

                perror("fork failed");            exit(1);

        case 0: {

                // Child process                  printf("This is the child process\n");
        printf("Running execve\n");             printf("The pid of the current child process:
%d\n", getpid());                 char *args[n + 2];                args[0] = "./hello";
        for (i = 0; i < n; i++) {

                                char *num = malloc(12);

                                snprintf(num, 12, "%d", arr[i]);

                                args[i + 1] = num;

                }

                        args[n + 1] = NULL;

  execve(args[0], args, NULL);    printf("Execve failed\n");
                        exit(1);
        }       default:
printf("This is the parent process\n");
wait(0);
printf("The pid of the current parent process: %d\n", getpid());
                        printf("\n");                       break;
```

```
        }
    return 0;
}
```

Hello.c

```
#include <stdio.h>
#include <stdlib.h> #include <unistd.h> int main(int argc, char *argv[]) {
    int n = argc - 1;     int arr[n];

for (int i = 0; i < n; i++) {
        arr[i] = atoi(argv[i + 1]);
    }
    printf("\nReversed array: ");
    for (int i = n - 1; i >= 0; i--) {        printf("%d ", arr[i]);
    }
    printf("\n");
    printf("The pid of the current process: %d\n", getpid());
    printf("\n");     return 0;
}
```

OUTPUT:

soham@soham-VirtualBox:~/Downloads$ gcc hello.c -o hello soham@soham-VirtualBox:~/Downloads$ gcc main.c -o main soham@soham-VirtualBox:~/Downloads$ ./main.out

Enter the number of elements: 5
Enter the elements:
4 3 1 5 2
Sorted array is
1 2 3 4 5
This is the parent process
The pid of the current parent process: 2345
Reversed array: 5 4 3 2 1
The pid of the current process: 2346