```c
/*
    Name : Ayush Gupta              Roll no : 33228
    Batch : L - 10                  Asgn : 2b
*/

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

    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++;
    }

    /* Copy the remaining elements of L[], if there
    are any */
    while (i < n1) {
        arr[k] = L[i];
```

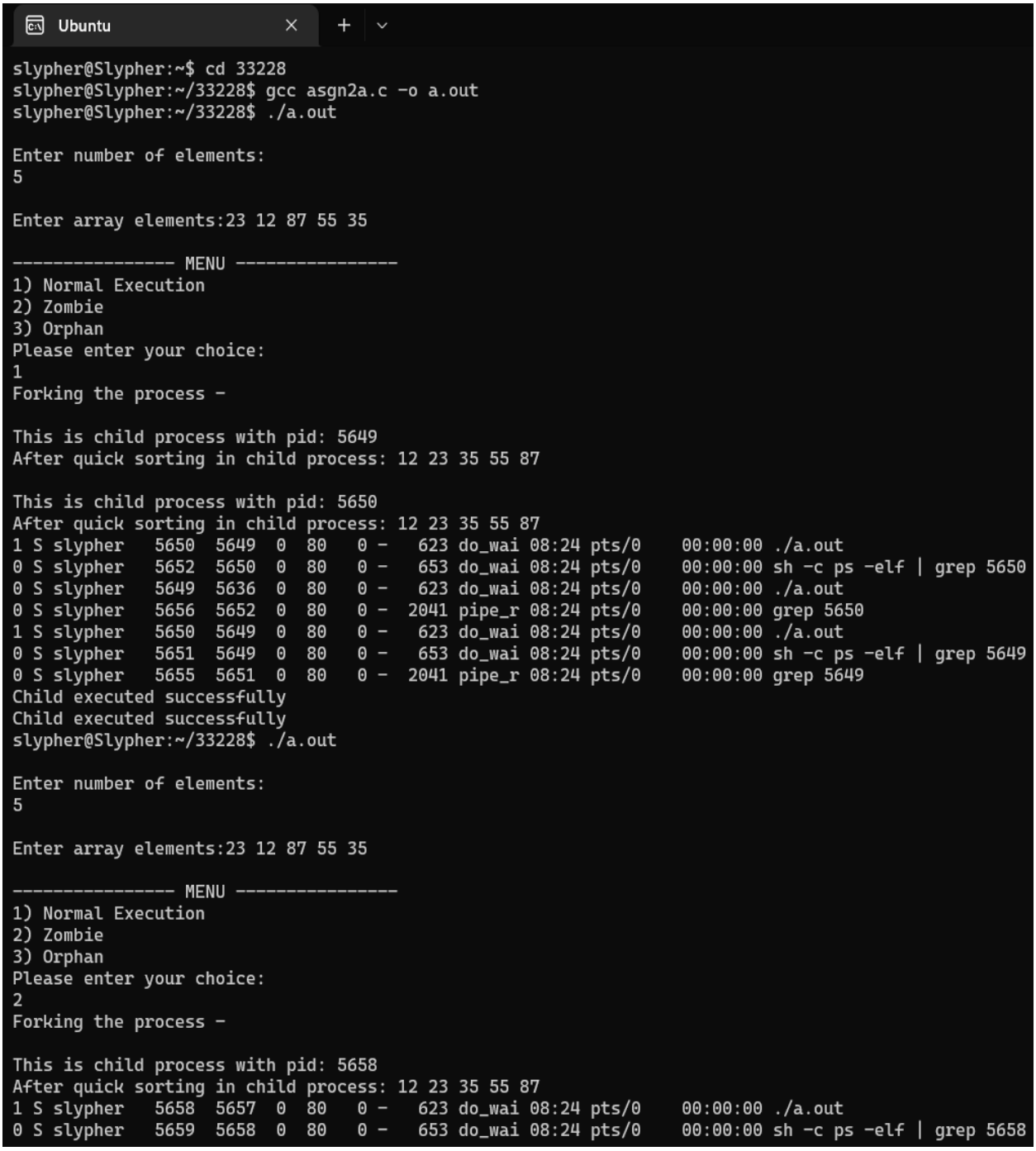
```
        i++;
        k++;
    }

    /* Copy the remaining elements of R[], if there
    are any */
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

/* l is for left index and r is right index of the
sub-array of arr to be sorted */
void mergeSort(int arr[], int l, int r)
{
    if (l < r) {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        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 swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}

// function to find the partition position
int partition(int array[], int low, int high) {

  // select the rightmost element as pivot
  int pivot = array[high];

  // pointer for greater element
  int i = (low - 1);
```

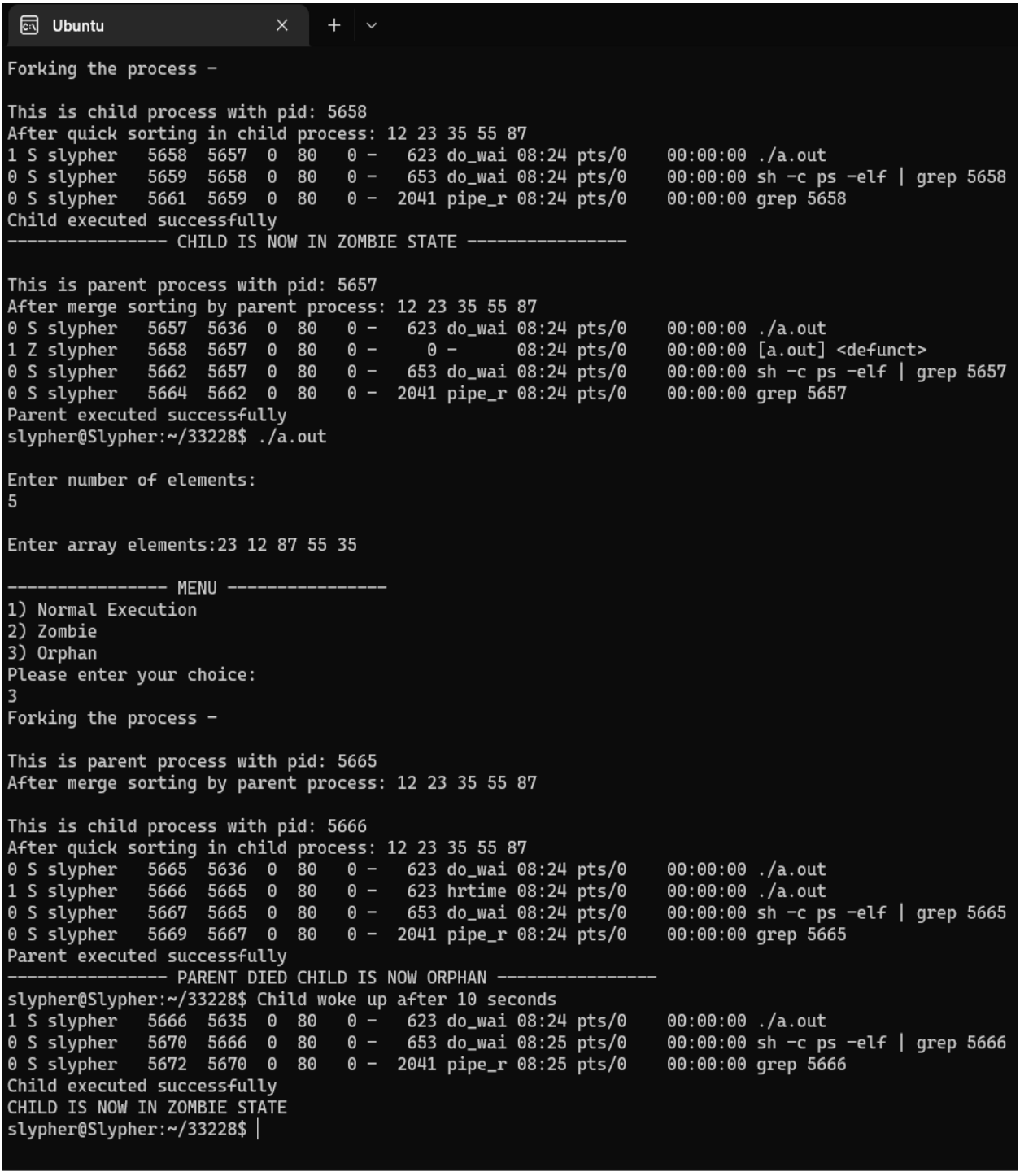
```
  // traverse each element of the array
  // compare them with the pivot
  for (int j = low; j < high; j++) {
    if (array[j] <= pivot) {

      // if element smaller than pivot is found
      // swap it with the greater element pointed by i
      i++;

      // swap element at i with element at j
      swap(&array[i], &array[j]);
    }
  }

  // swap the pivot element with the greater element at i
  swap(&array[i + 1], &array[high]);

  // return the partition point
  return (i + 1);
}

void quickSort(int array[], int low, int high) {
  if (low < high) {

    // find the pivot element such that
    // elements smaller than pivot are on left of pivot
    // elements greater than pivot are on right of pivot
    int pi = partition(array, low, high);

    // recursive call on the left of pivot
    quickSort(array, low, pi - 1);

    // recursive call on the right of pivot
    quickSort(array, pi + 1, high);
  }
}

int main() {
    char buffer[50];
    int a[10];
    int i, n, choice;
    pid_t x, cpid, newpid;
    printf("\nEnter number of elements: \n");
    scanf("%d", &n);
    printf("\nEnter array elements:");
```

```
for(i = 0; i < n; i++) {
    scanf("%d", &a[i]);
}
printf("\n---------------- MENU ----------------\n");
printf("1) Normal Execution\n");
printf("2) Zombie\n");
printf("3) Orphan\n");
printf("Please enter your choice: \n");
scanf("%d", &choice);

switch(choice) {
    case 1:
    // Fork System call
    printf("Forking the process - \n");
    newpid = fork();

    if(newpid == -1) {
        printf("Unfortunately child wasn't born");
    }
    /*Create child process */
    else if( x == 0) { // Child process
    printf("\nThis is child process with pid: %d", getpid());
    quickSort(a, 0, n - 1);
    printf("\nAfter quick sorting in child process: ");
    for(i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
    snprintf(buffer, 25, "ps -elf | grep %d", getpid());
    system(buffer);
    printf("Child executed successfully\n");
}

else { // Parent process
    printf("\nThis is parent process with pid: %d", getpid());
    cpid = wait(NULL);
    printf("\nChild is terminated after wait with pid: %d", cpid);
    mergeSort(a, 0, n - 1);
    printf("\nAfter merge sorting by parent process: ");
    for(i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    snprintf(buffer, 25, "ps -elf | grep %d", getpid());
    system(buffer);
    printf("Parent executed successfully\n");
```

```
        printf("\n");
    }
    break;

    case 2:
        // Zombieeeeeee
        // Fork System call
        printf("Forking the process - \n");
        newpid = fork();
        if(newpid == -1) {
            printf("Unfortunately child wasn't born");
        }
        else if(newpid == 0) {
            // Inside child process
            printf("\nThis is child process with pid: %d", getpid());
            quickSort(a, 0, n - 1);
            printf("\nAfter quick sorting in child process: ");
            for(i = 0; i < n; i++) {
                printf("%d ", a[i]);
            }
            printf("\n");
            snprintf(buffer, 25, "ps -elf | grep %d", getpid());
            system(buffer);
            printf("Child executed successfully\n");
            printf("---------------- CHILD IS NOW IN ZOMBIE STATE ---------------
-\n");

        }
        else { // Parent process
        sleep(10);
        printf("\nThis is parent process with pid: %d", getpid());
        mergeSort(a, 0, n - 1);
        printf("\nAfter merge sorting by parent process: ");
        for(i = 0; i < n; i++) {
            printf("%d ", a[i]);
        }
        printf("\n");
        snprintf(buffer, 25, "ps -elf | grep %d", getpid());
        system(buffer);
        printf("Parent executed successfully\n");
        wait(NULL);
        exit(0);
    }
    break;
```

```
    case 3:

        // Fork System call
        printf("Forking the process - \n");
        newpid = fork();
        if(newpid == -1) {
            printf("Unfortunately child wasn't born");
        }
        else if(newpid == 0) {
            // Inside child process
            printf("\nThis is child process with pid: %d", getpid());
            quickSort(a, 0, n - 1);
            printf("\nAfter quick sorting in child process: ");
            for(i = 0; i < n; i++) {
                printf("%d ", a[i]);
            }
            printf("\n");
            sleep(10);
            printf("Child woke up after 10 seconds\n");
            snprintf(buffer, 25, "ps -elf | grep %d", getpid());
            system(buffer);
            printf("Child executed successfully\n");
            printf("CHILD IS NOW IN ZOMBIE STATE\n");

        }
        else { // Parent process
        printf("\nThis is parent process with pid: %d", getpid());
        mergeSort(a, 0, n - 1);
        printf("\nAfter merge sorting by parent process: ");
        for(i = 0; i < n; i++) {
            printf("%d ", a[i]);
        }
        printf("\n");
        snprintf(buffer, 25, "ps -elf | grep %d", getpid());
        system(buffer);
        printf("Parent executed successfully\n");
        printf("---------------- PARENT DIED CHILD IS NOW ORPHAN ----------------
\n");
        exit(0);
    }

    }

}
```


Ubuntu
slypher@Slypher:~$ cd 33228
slypher@Slypher:~/33228$ gcc asgn2a.c -o a.out
slypher@Slypher:~/33228$ ./a.out
Enter number of elements:
5
Enter array elements:23 12 87 55 35
---------- MENU ----------
1) Normal Execution
2) Zombie
3) Orphan
Please enter your choice:
1
Forking the process -
This is child process with pid: 5649
After quick sorting in child process: 12 23 35 55 87
This is child process with pid: 5650
After quick sorting in child process: 12 23 35 55 87
1 S slypher 5650 5649 0 80 0 - 623 do_wai 08:24 pts/0 00:00:00 ./a.out
0 S slypher 5652 5650 0 80 0 - 653 do_wai 08:24 pts/0 00:00:00 sh -c ps -elf | grep 5650
0 S slypher 5649 5636 0 80 0 - 623 do_wai 08:24 pts/0 00:00:00 ./a.out
0 S slypher 5656 5652 0 80 0 - 2041 pipe_r 08:24 pts/0 00:00:00 grep 5650
1 S slypher 5650 5649 0 80 0 - 623 do_wai 08:24 pts/0 00:00:00 ./a.out
0 S slypher 5651 5649 0 80 0 - 653 do_wai 08:24 pts/0 00:00:00 sh -c ps -elf | grep 5649
0 S slypher 5655 5651 0 80 0 - 2041 pipe_r 08:24 pts/0 00:00:00 grep 5649
Child executed successfully
Child executed successfully
slypher@Slypher:~/33228$ ./a.out
Enter number of elements:
5
Enter array elements:23 12 87 55 35
---------- MENU ----------
1) Normal Execution
2) Zombie
3) Orphan
Please enter your choice:
2
Forking the process -
This is child process with pid: 5658
After quick sorting in child process: 12 23 35 55 87
1 S slypher 5658 5657 0 80 0 - 623 do_wai 08:24 pts/0 00:00:00 ./a.out
0 S slypher 5659 5658 0 80 0 - 653 do_wai 08:24 pts/0 00:00:00 sh -c ps -elf | grep 5658


Ubuntu
Forking the process -
This is child process with pid: 5658
After quick sorting in child process: 12 23 35 55 87
1 S slypher 5658 5657 0 80 0 - 623 do_wai 08:24 pts/0 00:00:00 ./a.out
0 S slypher 5659 5658 0 80 0 - 653 do_wai 08:24 pts/0 00:00:00 sh -c ps -elf | grep 5658
0 S slypher 5661 5659 0 80 0 - 2041 pipe_r 08:24 pts/0 00:00:00 grep 5658
Child executed successfully
---------- CHILD IS NOW IN ZOMBIE STATE ----------
This is parent process with pid: 5657
After merge sorting by parent process: 12 23 35 55 87
0 S slypher 5657 5636 0 80 0 - 623 do_wai 08:24 pts/0 00:00:00 ./a.out
1 Z slypher 5658 5657 0 80 0 - 0 - 08:24 pts/0 00:00:00 [a.out] <defunct>
0 S slypher 5662 5657 0 80 0 - 653 do_wai 08:24 pts/0 00:00:00 sh -c ps -elf | grep 5657
0 S slypher 5664 5662 0 80 0 - 2041 pipe_r 08:24 pts/0 00:00:00 grep 5657
Parent executed successfully
slypher@Slypher:~/33228$ ./a.out
Enter number of elements:
5
Enter array elements:23 12 87 55 35
---------- MENU ----------
1) Normal Execution
2) Zombie
3) Orphan
Please enter your choice:
3
Forking the process -
This is parent process with pid: 5665
After merge sorting by parent process: 12 23 35 55 87
This is child process with pid: 5666
After quick sorting in child process: 12 23 35 55 87
0 S slypher 5665 5636 0 80 0 - 623 do_wai 08:24 pts/0 00:00:00 ./a.out
1 S slypher 5666 5665 0 80 0 - 623 hrtime 08:24 pts/0 00:00:00 ./a.out
0 S slypher 5667 5665 0 80 0 - 653 do_wai 08:24 pts/0 00:00:00 sh -c ps -elf | grep 5665
0 S slypher 5669 5667 0 80 0 - 2041 pipe_r 08:24 pts/0 00:00:00 grep 5665
Parent executed successfully
---------- PARENT DIED CHILD IS NOW ORPHAN ----------
slypher@Slypher:~/33228$ Child woke up after 10 seconds
1 S slypher 5666 5635 0 80 0 - 623 do_wai 08:24 pts/0 00:00:00 ./a.out
0 S slypher 5670 5666 0 80 0 - 653 do_wai 08:25 pts/0 00:00:00 sh -c ps -elf | grep 5666
0 S slypher 5672 5670 0 80 0 - 2041 pipe_r 08:25 pts/0 00:00:00 grep 5666
Child executed successfully
CHILD IS NOW IN ZOMBIE STATE
slypher@Slypher:~/33228$