Assignment2B:-

Child.c

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

int main(int argc, char *argv[]) {
  if (argc < 2) {
      printf("No array elements received.\n");
      return 1;
  }

  printf("Child process started.\n");

  printf("Array in reverse order:\n");
  for (int i = argc - 1; i >= 1; i--) {
      printf("%s ", argv[i]);
  }
  printf("\n");

  printf("Child process finished.\n");

  return 0;
}
```

Parent.c

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

int cmp(const void *a, const void *b) {
  return (*(int*)a - *(int*)b);
}

int main() {
  int n;
  printf("Enter the number of elements: ");
  scanf("%d", &n);
  int arr[n];

  printf("Enter all elements:\n");
  for (int i = 0; i < n; i++)
      scanf("%d", &arr[i]);

  qsort(arr, n, sizeof(int), cmp);

  pid_t pid = fork();

  if (pid > 0) {
      // Parent waits for child to complete
      wait(NULL);
  }
  else if (pid == 0) {
      // Child prepares args and execve

      char *args[n + 2];
      args[0] = "./child";

      for (int i = 0; i < n; i++) {
          args[i + 1] = malloc(12);
          sprintf(args[i + 1], "%d", arr[i]);
      }
      args[n + 1] = NULL;

      execve("./child", args, NULL);

      // execve only returns on failure
      perror("execve failed");

      for (int i = 1; i <= n; i++) free(args[i]);
      exit(1);
  }
  else {
      perror("fork failed");
      return 1;
  }
  return 0;
}
```

OUTPUT:-

```
pict@mplab-12:~/Desktop/33164$ gcc parent.c -o parent
pict@mplab-12:~/Desktop/33164$ gcc child.c -o child
pict@mplab-12:~/Desktop/33164$ ./parent
Enter the number of elements: 5
Enter all elements:
1
2
3
4
5
Child process started.
Array in reverse order:
5 4 3 2 1
Child process finished.
pict@mplab-12:~/Desktop/33164$
```