;

OPERATING SYSTEMS LAB

PRACTICAL 6

NAME: VEDANT BHUTADA

ROLL: 69

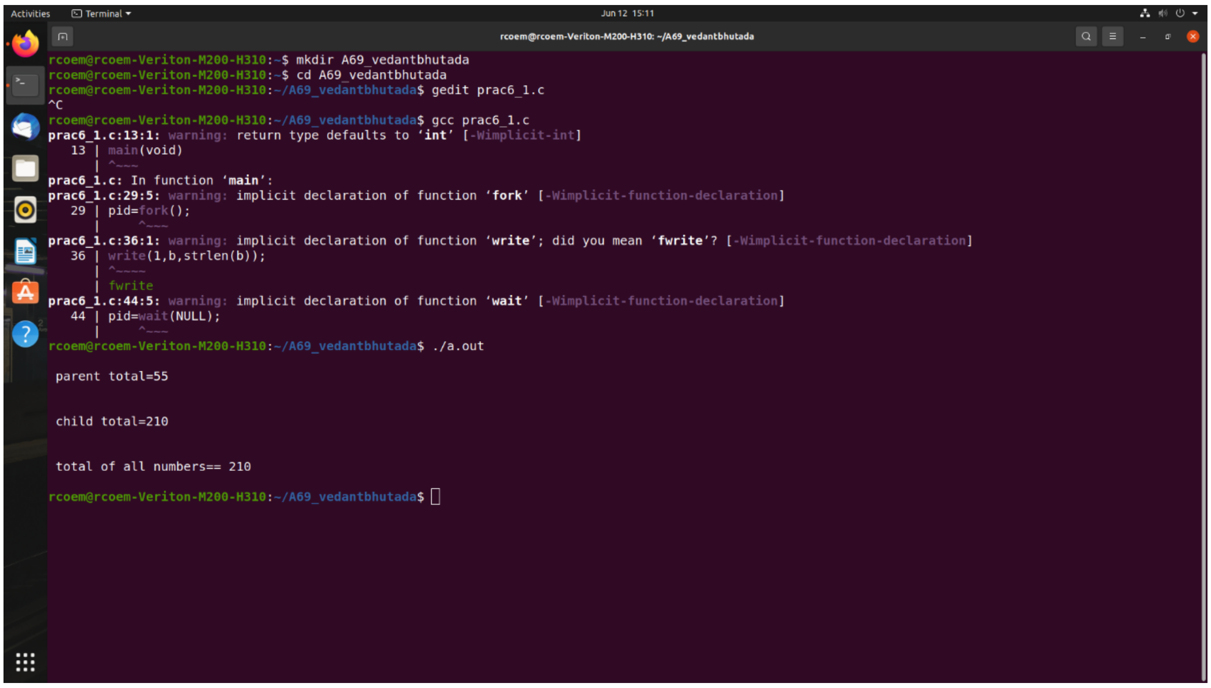
BATCH: A4

AIM: Implement a C program to demonstrate the concept of Shared Memory.

CODE:

1) shared memory basic program to find the total of n numbers.

```
#include<stdio.h>

#include<string.h>

#include<fcntl.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<sys/shm.h>



#define buf_size 100


int a[]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20};


main(void)

{


pid_t pid;

int i;

int *total;

char b[buf_size];
//get the segment//
int segment_id=shmget(IPC_PRIVATE,2,S_IRUSR|S_IWUSR);

//attach the segment with variable to be used by process
total=(int*)shmat(segment_id,NULL,0);
*total=0;

//creat new child//
pid=fork();
if(pid==0)
{

for(i=10;i<20;i++)
*total= *total + a[i];
sprintf(b,"\n child total=%d \n\n",*total);
write(1,b,strlen(b));
}
else
{
for(i=0;i<10;i++)
*total= *total + a[i];
sprintf(b,"\n parent total=%d \n\n",*total);
write(1,b,strlen(b));
pid=wait(NULL);
if(pid!=-1)
printf("\n total of all numbers== %d\n\n",*total);
shmdt(total);
shmctl(segment_id, IPC_RMID, NULL);

} }
```



```
rcoem@rcoem-Veriton-M200-H310:~$ mkdir A69_vedantbhutada
rcoem@rcoem-Veriton-M200-H310:~$ cd A69_vedantbhutada
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ gedit prac6_1.c
^C
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ gcc prac6_1.c
prac6_1.c:13:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   13 | main(void)
      | ^~~~
prac6_1.c: In function ‘main’:
prac6_1.c:29:5: warning: implicit declaration of function ‘fork’ [-Wimplicit-function-declaration]
   29 | pid=fork();
      |     ^~~~
prac6_1.c:36:1: warning: implicit declaration of function ‘write’; did you mean ‘fwrite’? [-Wimplicit-function-declaration]
   36 | write(1,b,strlen(b));
      | ^~~~~
      | fwrite
prac6_1.c:44:5: warning: implicit declaration of function ‘wait’ [-Wimplicit-function-declaration]
   44 | pid=wait(NULL);
      |     ^~~~
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ ./a.out

 parent total=55


 child total=210


 total of all numbers== 210

rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ 
```

---

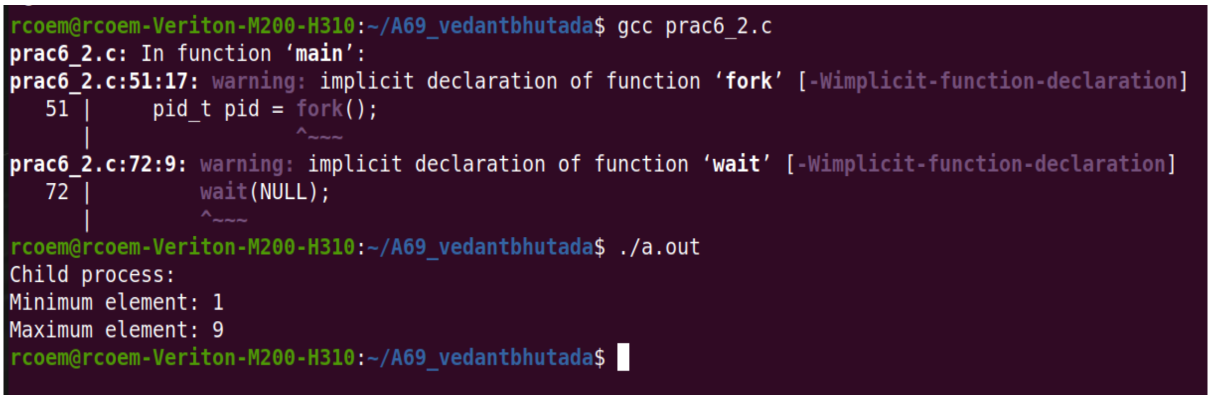
2) To find the maximum and minimum element in an array using shared memory.

```
#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>


#define SHM_SIZE 100


void findMinMax(int* array, int size, int* min, int* max) {

    *min = array[0];

    *max = array[0];


    for (int i = 1; i < size; i++) {

        if (array[i] < *min) {

            *min = array[i];
```

```
        }

        if (array[i] > *max) {
            *max = array[i];
        }
    }
}

int main() {
    int array[] = {4, 2, 9, 1, 7, 5};
    int size = sizeof(array) / sizeof(array[0]);

    // Create shm
    key_t key = IPC_PRIVATE;
    int shm_id = shmget(key, SHM_SIZE, IPC_CREAT | 0666);

    if (shm_id < 0) {
        perror("shmget");
        exit(1);
    }

    // Attach the shm to the process
    int* shm_ptr = (int*)shmat(shm_id, NULL, 0);

    if (shm_ptr == (int*)-1) {
        perror("shmat");
        exit(1);
    }

    // Copying the array to shared memory
    for (int i = 0; i < size; i++) {
```

```
    shm_ptr[i] = array[i];
}

// Fork a child process
pid_t pid = fork();

if (pid < 0) {
    perror("fork");
    exit(1);
}

if (pid == 0) {
    // Child process
    int min, max;
    findMinMax(shm_ptr, size, &min, &max);
    printf("Child process:\n");
    printf("Minimum element: %d\n", min);
    printf("Maximum element: %d\n", max);

    // Detach the shm
    shmdt(shm_ptr);

    exit(0);
} else {
    // Parent process
    wait(NULL);

    // Detach and remove the shm
    shmdt(shm_ptr);
    shmctl(shm_id, IPC_RMID, NULL);
}
```

```
    return 0;

}
```



```
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ gcc prac6_2.c
prac6_2.c: In function ‘main’:
prac6_2.c:51:17: warning: implicit declaration of function ‘fork’ [-Wimplicit-function-declaration]
   51 |     pid_t pid = fork();
      |                 ^~~~
prac6_2.c:72:9: warning: implicit declaration of function ‘wait’ [-Wimplicit-function-declaration]
   72 |         wait(NULL);
      |         ^~~~
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ ./a.out
Child process:
Minimum element: 1
Maximum element: 9
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ 
```

---

3) Example two processes comunicating via shared memory: shm_server.c, shm_client.c

Server.c

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

#include <sys/ipc.h>

#include <sys/shm.h>

#include <stdio.h>


#define SHMSZ     27


main()

{

    char c;

    int shmid;

    key_t key;

    char *shm, *s;


    /*
```

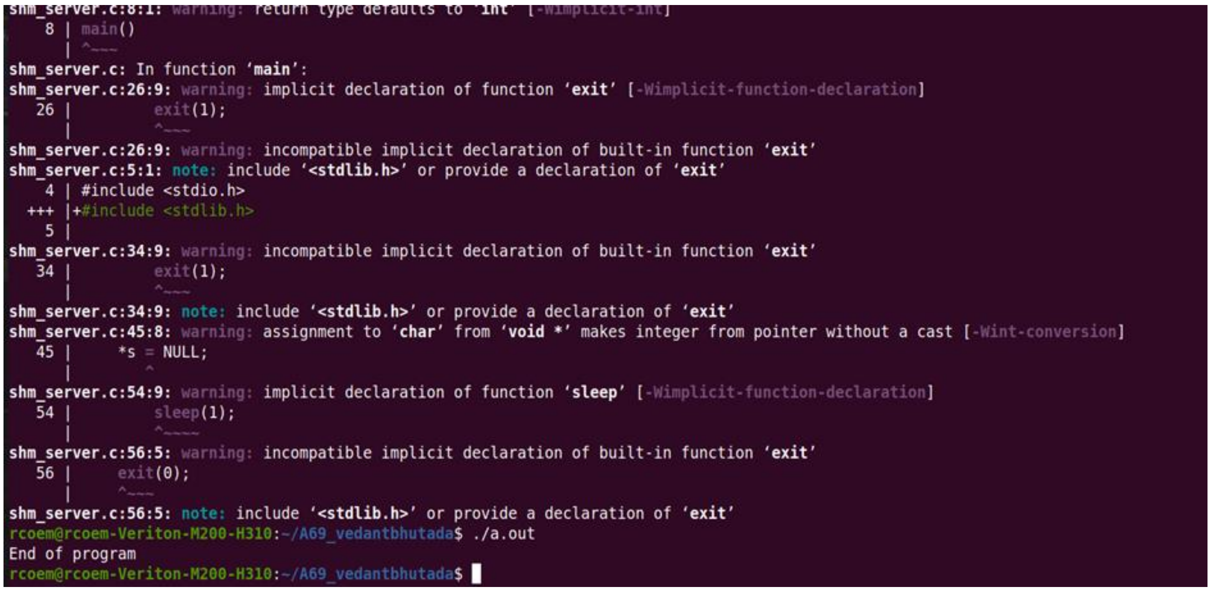
```
 * We'll name our shared memory segment
 * "5678".
 */
key = 5678;

/*
 * Create the segment.
 */
if ((shmid = shmget(key, SHMSZ, IPC_CREAT | 0666)) < 0) {
    perror("shmget");
    exit(1);
}

/*
 * Now we attach the segment to our data space.
 */
if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {
    perror("shmat");
    exit(1);
}

/*
 * Now put some things into the memory for the
 * other process to read.
 */
s = shm;

for (c = 'a'; c <= 'z'; c++)
    *s++ = c;
*s = NULL;
```

```
   /*
    * Finally, we wait until the other process
    * changes the first character of our memory
    * to '*', indicating that it has read what
    * we put there.
    */
   while (*shm != '*')
       sleep(1);
 printf("End of program\n");
   exit(0);
}
```

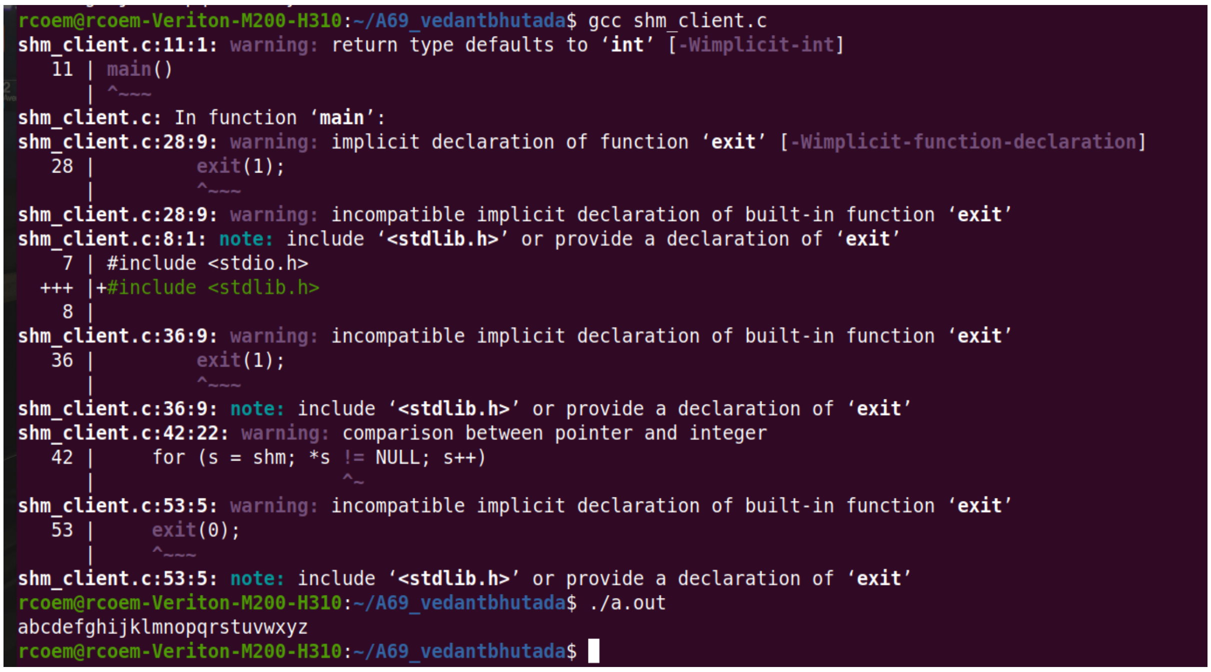
Client.c

```
/*
 * shm-client - client program to demonstrate shared memory.
 */
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>

#define SHMSZ     27

main()
{
   int shmid;
   key_t key;
   char *shm, *s;

   /*
    * We need to get the segment named
 * "5678", created by the server.
 */
key = 5678;

/*
 * Locate the segment.
 */
if ((shmid = shmget(key, SHMSZ, 0666)) < 0) {
    perror("shmget");
    exit(1);
}

/*
 * Now we attach the segment to our data space.
 */
if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {
    perror("shmat");
    exit(1);
}

/*
 * Now read what the server put in the memory.
 */
for (s = shm; *s != NULL; s++)
    putchar(*s);
putchar('\n');

/*
 * Finally, change the first character of the
 * segment to '*', indicating we have read
 * the segment.
```

```
  */
  *shm = '*';


  exit(0);
}
```



```
shm_server.c:8:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
    8 | main()
      | ^~~~
shm_server.c: In function ‘main’:
shm_server.c:26:9: warning: implicit declaration of function ‘exit’ [-Wimplicit-function-declaration]
   26 |         exit(1);
      |         ^~~~
shm_server.c:26:9: warning: incompatible implicit declaration of built-in function ‘exit’
shm_server.c:5:1: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
    4 | #include <stdio.h>
  +++ |+#include <stdlib.h>
    5 |
shm_server.c:34:9: warning: incompatible implicit declaration of built-in function ‘exit’
   34 |         exit(1);
      |         ^~~~
shm_server.c:34:9: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
shm_server.c:45:8: warning: assignment to ‘char’ from ‘void *’ makes integer from pointer without a cast [-Wint-conversion]
   45 |     *s = NULL;
      |        ^
shm_server.c:54:9: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function-declaration]
   54 |         sleep(1);
      |         ^~~~~
shm_server.c:56:5: warning: incompatible implicit declaration of built-in function ‘exit’
   56 |     exit(0);
      |     ^~~~
shm_server.c:56:5: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ ./a.out
End of program
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ 
```



```
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ gcc shm_client.c
shm_client.c:11:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   11 | main()
      | ^~~~
shm_client.c: In function ‘main’:
shm_client.c:28:9: warning: implicit declaration of function ‘exit’ [-Wimplicit-function-declaration]
   28 |         exit(1);
      |         ^~~~
shm_client.c:28:9: warning: incompatible implicit declaration of built-in function ‘exit’
shm_client.c:8:1: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
    7 | #include <stdio.h>
  +++ |+#include <stdlib.h>
    8 |
shm_client.c:36:9: warning: incompatible implicit declaration of built-in function ‘exit’
   36 |         exit(1);
      |         ^~~~
shm_client.c:36:9: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
shm_client.c:42:22: warning: comparison between pointer and integer
   42 |     for (s = shm; *s != NULL; s++)
      |                      ^~
shm_client.c:53:5: warning: incompatible implicit declaration of built-in function ‘exit’
   53 |     exit(0);
      |     ^~~~
shm_client.c:53:5: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ ./a.out
abcdefghijklmnopqrstuvwxyz
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ 
```

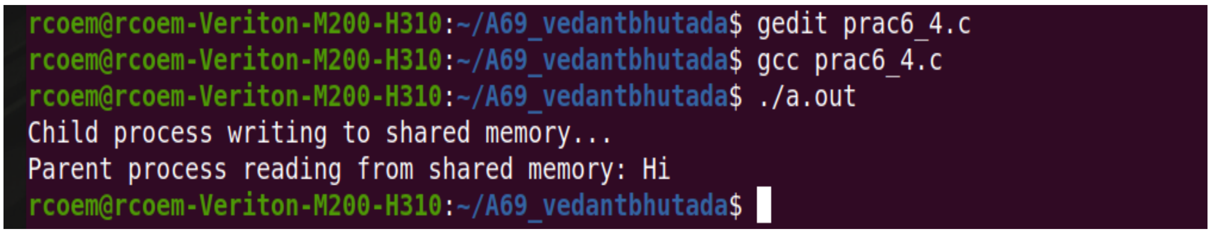
---

4) Write a C program that illustrates 2 processes communicating using shared memory.

```
#include <stdio.h>

#include <stdlib.h>
```

```
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/wait.h>

#define SHM_SIZE 1024

int main() {
  int shmid;
  key_t key;
  char *shm, *s;

  // Generate a unique key for shared memory
  key = ftok(".", 'S');
  if (key == -1) {
    perror("ftok");
    exit(1);
  }

  // Create shared memory segment
  shmid = shmget(key, SHM_SIZE, IPC_CREAT | 0666);
  if (shmid == -1) {
    perror("shmget");
    exit(1);
  }

  // Attach the shared memory segment
  shm = shmat(shmid, NULL, 0);
  if (shm == (char *) -1) {
    perror("shmat");
    exit(1);
```

```
}

// Fork a child process
pid_t pid = fork();
if (pid < 0) {
    perror("fork");
    exit(1);
}

if (pid == 0) {
    // Child process
    printf("Child process writing to shared memory...\n");
    s = shm;
    *s = 'H';  // Write a character to shared memory
    s++;       // Move to the next position
    *s = 'i';  // Write another character to shared memory
    s++;
    *s = '\0'; // Terminate the string in shared memory
    exit(0);
} else {
    // Parent process
    wait(NULL); // Wait for child process to complete
    printf("Parent process reading from shared memory: %s\n", shm);
}

// Detach and remove shared memory segment
if (shmdt(shm) == -1) {
    perror("shmdt");
    exit(1);
}
if (shmctl(shmid, IPC_RMID, 0) == -1) {
        perror("shmctl");
        exit(1);
    }

    return 0;
}
```

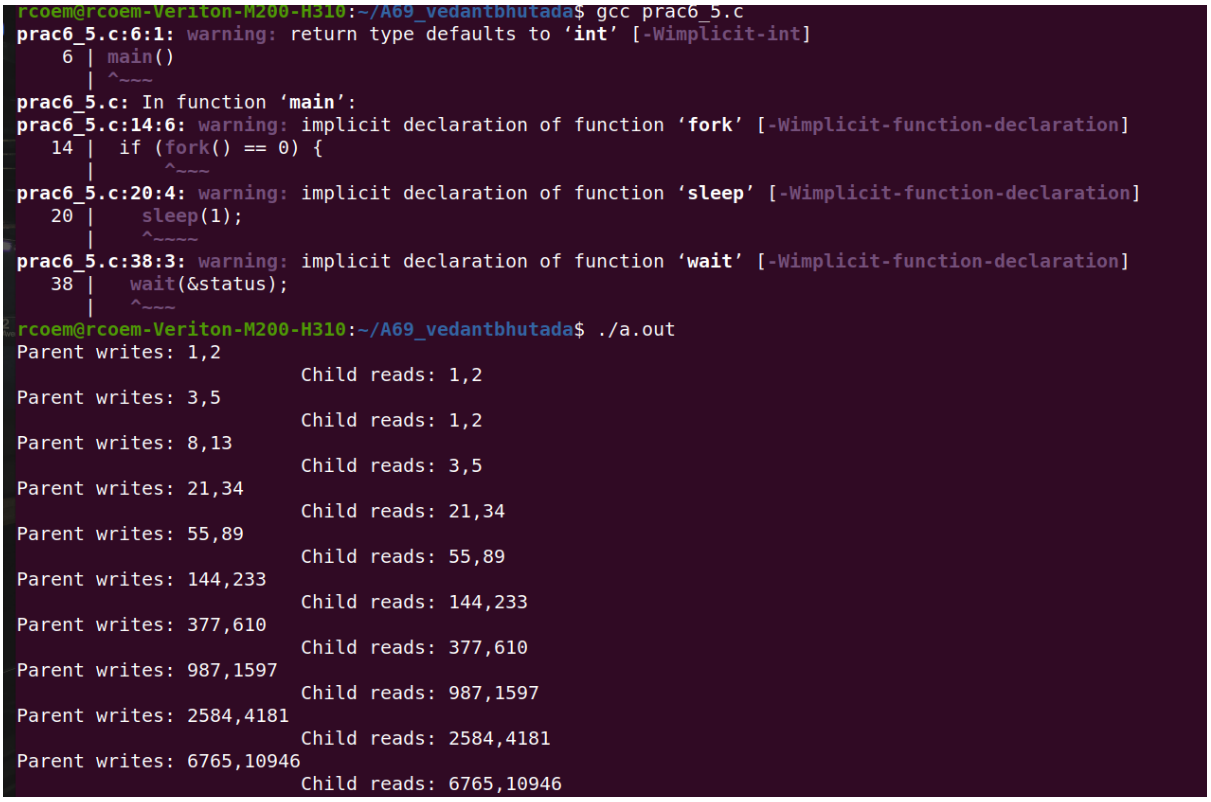


```
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ gedit prac6_4.c
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ gcc prac6_4.c
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ ./a.out
Child process writing to shared memory...
Parent process reading from shared memory: Hi
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ 
```

---

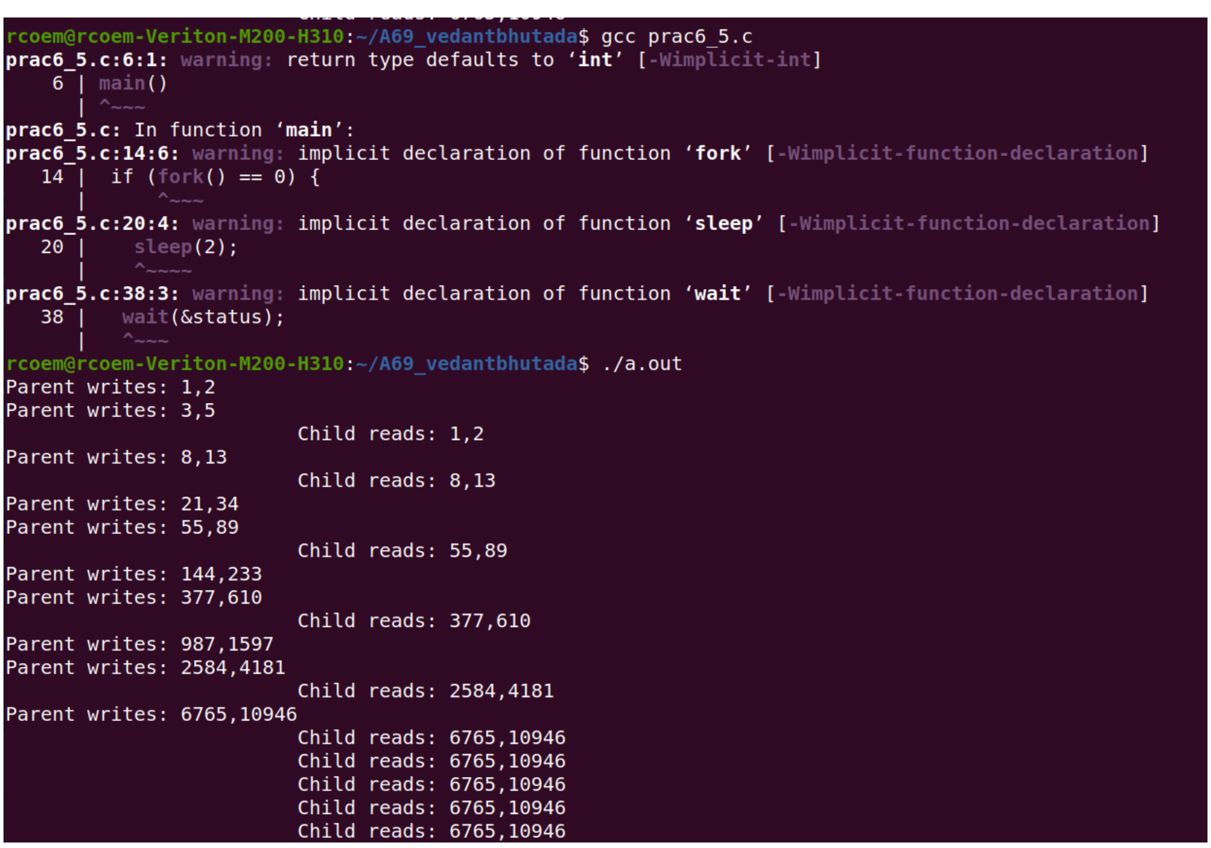
5) SHARING MEMORY BETWEEN PROCESSES

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/shm.h>

main()
{
        int shmid, status;
        int *a, *b;
        int i;

        shmid = shmget(IPC_PRIVATE, 2*sizeof(int), 0777|IPC_CREAT);

        if (fork() == 0) {

                /* Child Process */
                b = (int *) shmat(shmid, 0, 0);
                for( i=0; i< 10; i++) {
                        sleep(1);
                        printf("\t\t\t Child reads: %d,%d\n",b[0],b[1]);
                }
                shmdt(b);

        }
        else {

                /* Parent Process */
                a = (int *) shmat(shmid, 0, 0);

                a[0] = 0; a[1] = 1;
                for( i=0; i< 10; i++) {
                        sleep(2);
                        a[0] = a[0] + a[1];
                        a[1] = a[0] + a[1];
                        printf("Parent writes: %d,%d\n",a[0],a[1]);
                }
                wait(&status);
        shmdt(a);
                shmctl(shmid, IPC_RMID, 0);
        }
}
```



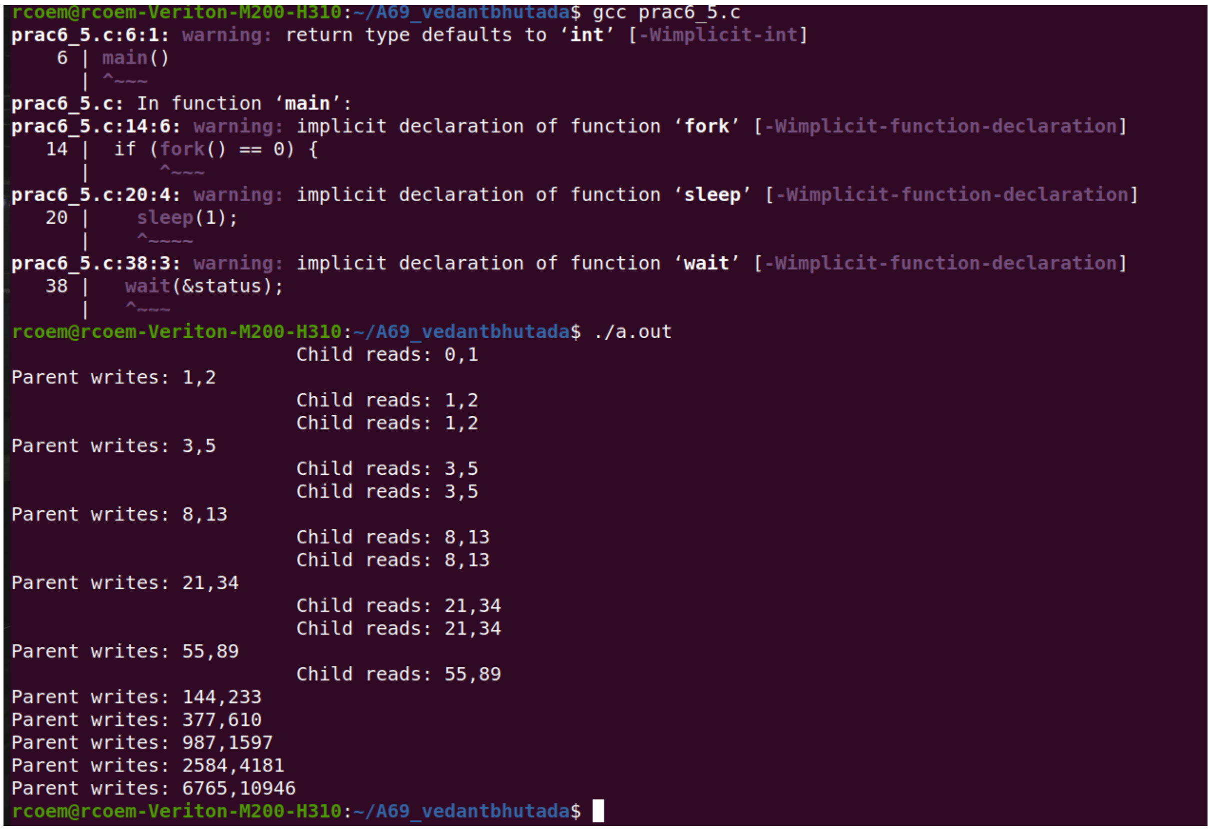
```
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ gcc prac6_5.c
prac6_5.c:6:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
    6 | main()
      | ^~~~
prac6_5.c: In function ‘main’:
prac6_5.c:14:6: warning: implicit declaration of function ‘fork’ [-Wimplicit-function-declaration]
   14 |  if (fork() == 0) {
      |      ^~~~
prac6_5.c:20:4: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function-declaration]
   20 |    sleep(1);
      |    ^~~~~
prac6_5.c:38:3: warning: implicit declaration of function ‘wait’ [-Wimplicit-function-declaration]
   38 |   wait(&status);
      |   ^~~~
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ ./a.out
Parent writes: 1,2
                         Child reads: 1,2
Parent writes: 3,5
                         Child reads: 1,2
Parent writes: 8,13
                         Child reads: 3,5
Parent writes: 21,34
                         Child reads: 21,34
Parent writes: 55,89
                         Child reads: 55,89
Parent writes: 144,233
                         Child reads: 144,233
Parent writes: 377,610
                         Child reads: 377,610
Parent writes: 987,1597
                         Child reads: 987,1597
Parent writes: 2584,4181
                         Child reads: 2584,4181
Parent writes: 6765,10946
                         Child reads: 6765,10946
```

Child sleep(2)



```
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ gcc prac6_5.c
prac6_5.c:6:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
    6 | main()
      | ^~~~
prac6_5.c: In function ‘main’:
prac6_5.c:14:6: warning: implicit declaration of function ‘fork’ [-Wimplicit-function-declaration]
   14 |  if (fork() == 0) {
      |      ^~~~
prac6_5.c:20:4: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function-declaration]
   20 |    sleep(2);
      |    ^~~~~
prac6_5.c:38:3: warning: implicit declaration of function ‘wait’ [-Wimplicit-function-declaration]
   38 |   wait(&status);
      |   ^~~~
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ ./a.out
Parent writes: 1,2
Parent writes: 3,5
                         Child reads: 1,2
Parent writes: 8,13
                         Child reads: 8,13
Parent writes: 21,34
Parent writes: 55,89
                         Child reads: 55,89
Parent writes: 144,233
Parent writes: 377,610
                         Child reads: 377,610
Parent writes: 987,1597
Parent writes: 2584,4181
                         Child reads: 2584,4181
Parent writes: 6765,10946
                         Child reads: 6765,10946
                         Child reads: 6765,10946
                         Child reads: 6765,10946
                         Child reads: 6765,10946
                         Child reads: 6765,10946
```

Parent sleep(2)



```
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ gcc prac6_5.c
prac6_5.c:6:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
    6 | main()
      | ^~~~
prac6_5.c: In function ‘main’:
prac6_5.c:14:6: warning: implicit declaration of function ‘fork’ [-Wimplicit-function-declaration]
   14 |  if (fork() == 0) {
      |      ^~~~
prac6_5.c:20:4: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function-declaration]
   20 |    sleep(1);
      |    ^~~~~
prac6_5.c:38:3: warning: implicit declaration of function ‘wait’ [-Wimplicit-function-declaration]
   38 |   wait(&status);
      |   ^~~~
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$ ./a.out
                         Child reads: 0,1
Parent writes: 1,2
                         Child reads: 1,2
                         Child reads: 1,2
Parent writes: 3,5
                         Child reads: 3,5
                         Child reads: 3,5
Parent writes: 8,13
                         Child reads: 8,13
                         Child reads: 8,13
Parent writes: 21,34
                         Child reads: 21,34
                         Child reads: 21,34
Parent writes: 55,89
                         Child reads: 55,89
Parent writes: 144,233
Parent writes: 377,610
Parent writes: 987,1597
Parent writes: 2584,4181
Parent writes: 6765,10946
rcoem@rcoem-Veriton-M200-H310:~/A69_vedantbhutada$
```

---

6) Use of shared memory
Part 1
Write 2 C programs A.c and B.c. The program A.c reads in a set of integers (maximum 100) from the file named
inpfile and writes it to a shared array. The program B.c reads the set of integers from the shared array (how will
it know how many integers are there?), sorts it and prints the sorted output in a file named outfile.
Make sure that B.c deletes all shared memory created before exiting.
There is an obvious synchronization problem here, B.c should not start until A.c has finished writing the integers
in the array. For the first part, ignore it, and start the program for B.c a few seconds after A.c starts.
Part 2
In this part, we will try to synchronize A.c and B.c by a simple method. Create a shared integer variable called
done and initialize it to 0. done = 0 indicates that A.c has not finished writing the integers into the array. The
program in A.c sets done to 1 after it finishes. The program in B.c periodically checks done and loops until it is
Modify A.c and B.c to implement this.

A_1.c

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

```c
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>

#define MAX_INTS 100

int main() {
   FILE *inpfile;
   int sharedMemoryId;
   int *sharedArray;
   int numInts = 0;

   // Create shared memory
   sharedMemoryId = shmget(IPC_PRIVATE, MAX_INTS * sizeof(int), IPC_CREAT | 0666);
   if (sharedMemoryId == -1) {
      printf("Error creating shared memory.\n");
      return 1;
   }

   // Attach shared memory to the process
   sharedArray = (int *)shmat(sharedMemoryId, NULL, 0);
   if (sharedArray == (int *)(-1)) {
      printf("Error attaching shared memory.\n");
      return 1;
   }

   inpfile = fopen("inpfile.txt", "r");
   if (inpfile == NULL) {
      printf("Error opening inpfile.\n");
      return 1;
   }

   // Read integers from inpfile into sharedArray
   while (fscanf(inpfile, "%d", &sharedArray[numInts]) != EOF && numInts < MAX_INTS) {
      numInts++;
   }

   fclose(inpfile);

   // Detach shared memory from the process
   shmdt(sharedArray);

   sleep(5); // Wait for B.c to start after A.c has finished writing to shared memory

   // Delete shared memory
   shmctl(sharedMemoryId, IPC_RMID, NULL);

   return 0;
}
```

B_1.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define MAX_INTS 100

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

int main() {
    int sharedMemoryId;
    int *sharedArray;
    int numInts = 0;

    // Attach to shared memory created by A.c
    sharedMemoryId = shmget(IPC_PRIVATE, MAX_INTS * sizeof(int), 0666);
    if (sharedMemoryId == -1) {
        printf("Error accessing shared memory.\n");
        return 1;
    }

    // Attach shared memory to the process
    sharedArray = (int *)shmat(sharedMemoryId, NULL, 0);
    if (sharedArray == (int *)(-1)) {
        printf("Error attaching shared memory.\n");
        return 1;
    }
    // Read integers from sharedArray

    while (sharedArray[numInts] != 0 && numInts < MAX_INTS) {

        numInts++;

    }


    // Sort the integers

    qsort(sharedArray, numInts, sizeof(int), cmpfunc);


    // Create outfile to write the sorted integers

    FILE *outfile = fopen("outfile", "w");

    if (outfile == NULL) {

        printf("Error creating outfile.\n");

        return 1;

    }


    // Write the sorted integers to outfile

    for (int i = 0; i < numInts; i++) {

        fprintf(outfile, "%d\n", sharedArray[i]);

    }


    fclose(outfile);


    // Detach shared memory from the process

    shmdt(sharedArray);


    return 0;

}
```

A_2.c

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

```
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>
#define MAX_INTS 100

int main() {
  FILE *inpfile;
  int sharedMemoryId;
  int *sharedArray;
  int numInts = 0;

  // Create shared memory
  sharedMemoryId = shmget(IPC_PRIVATE, MAX_INTS * sizeof(int), IPC_CREAT | 0666);
  if (sharedMemoryId == -1) {
    printf("Error creating shared memory.\n");
    return 1;
  }

  // Attach shared memory to the process
  sharedArray = (int *)shmat(sharedMemoryId, NULL, 0);
  if (sharedArray == (int *)(-1)) {
    printf("Error attaching shared memory.\n");
    return 1;
  }

  inpfile = fopen("inpfile.txt", "r");
  if (inpfile == NULL) {
    printf("Error opening inpfile.\n");
    return 1;
  }
```

```
    // Read integers from inpfile into sharedArray
    while (fscanf(inpfile, "%d", &sharedArray[numInts]) != EOF) {
        numInts++;
    }

    fclose(inpfile);

    // Set done to 1 to indicate A.c has finished writing
    sharedArray[MAX_INTS - 1] = 1;

    // Detach shared memory from the process
    shmdt(sharedArray);

    return 0;
}
```

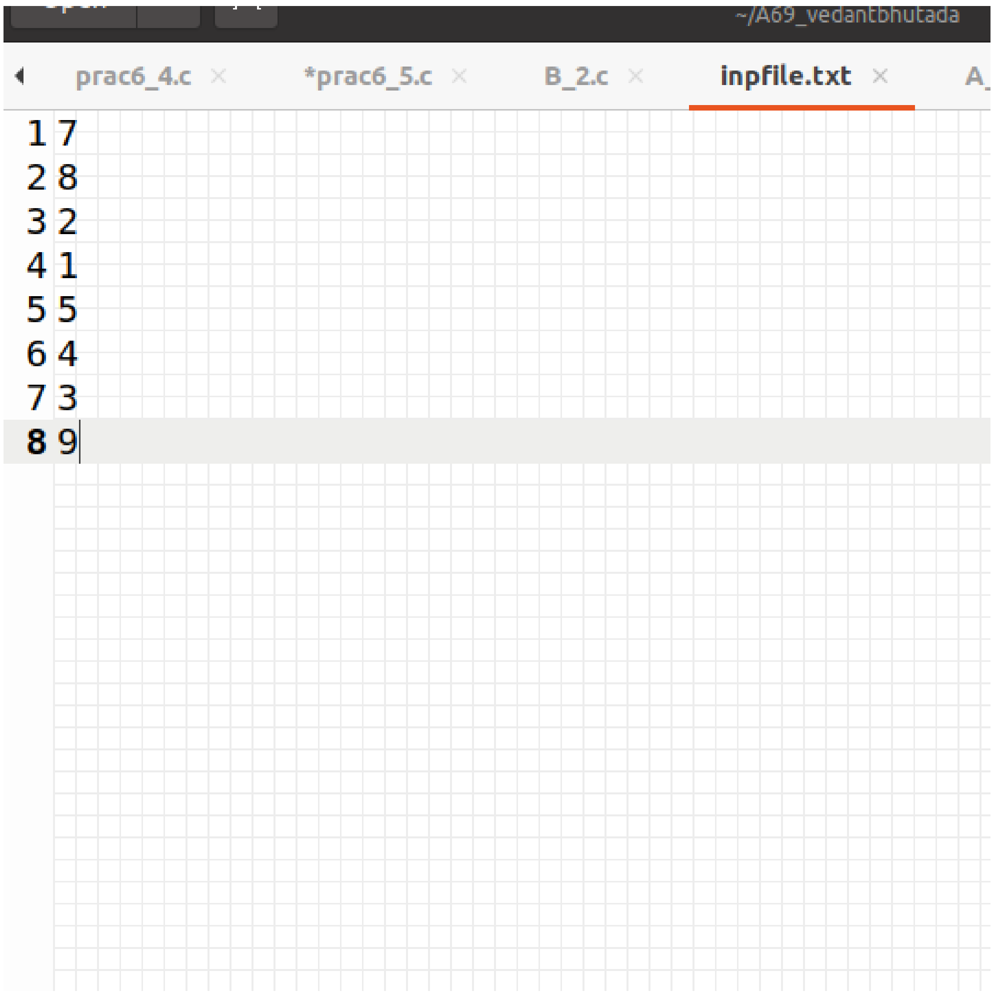
B_2.c

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

#define MAX_INTS 100

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

int main() {
```

```
int sharedMemoryId;
int *sharedArray;
int numInts = 0;
int done = 0;

// Create shared memory
sharedMemoryId = shmget(IPC_PRIVATE, MAX_INTS * sizeof(int), IPC_CREAT | 0666);
if (sharedMemoryId == -1) {
    printf("Error creating shared memory.\n");
    return 1;
}

// Attach shared memory to the process
sharedArray = (int *)shmat(sharedMemoryId, NULL, 0);
if (sharedArray == (int *)(-1)) {
    printf("Error attaching shared memory.\n");
    return 1;
}

// Periodically check if A.c has finished writing
while (!done) {
    done = sharedArray[MAX_INTS - 1];
    sleep(1);
}

// Read integers from sharedArray
for (int i = 0; i < MAX_INTS - 1; i++) {
    if (sharedArray[i] != 0) {
        numInts++;
    }
}
```

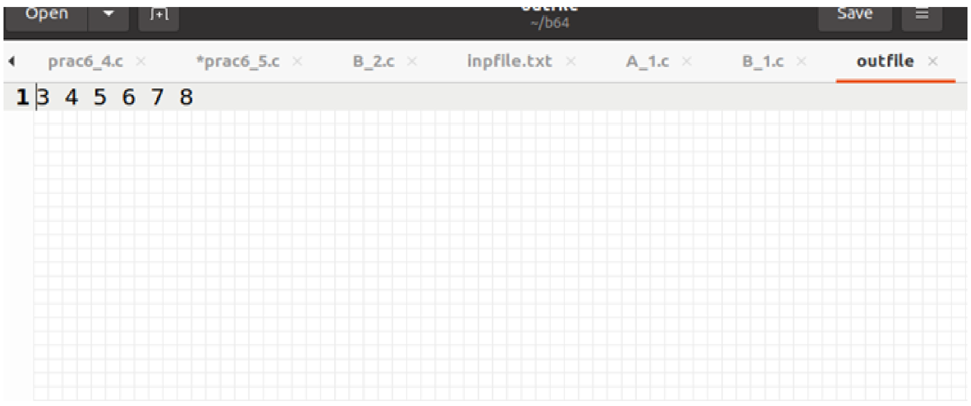
```
    // Sort the integers
    qsort(sharedArray, numInts, sizeof(int), cmpfunc);

    // Create outfile to write the sorted integers
    FILE *outfile = fopen("outfile", "w");
    if (outfile == NULL) {
        printf("Error creating outfile.\n");
        return 1;
    }

    // Write the sorted integers to outfile
    for (int i = 0; i < numInts; i++) {
        fprintf(outfile, "%d\n", sharedArray[i]);
    }

    fclose(outfile);

    // Delete shared memory
    shmctl(sharedMemoryId, IPC_RMID, NULL);

    return 0;
}
```

INPUT FILE



```
7
8
2
1
5
4
3
9
```

OUTPUT FILE



```
3  4  5  6  7  8
```

Result: Linux C programs to demonstrate the concept of Shared Memory has been implemented.