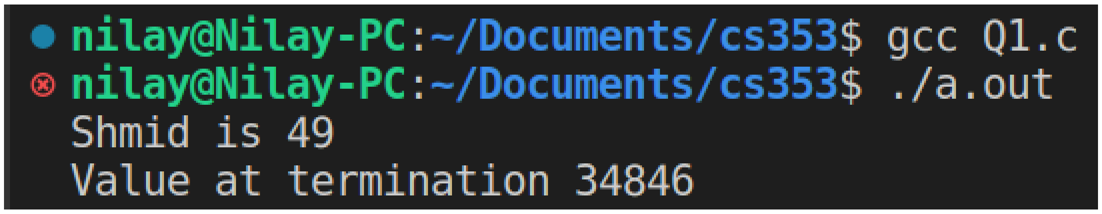
CS353
Lab5
200001053
Nilay Ganvit

Q1.

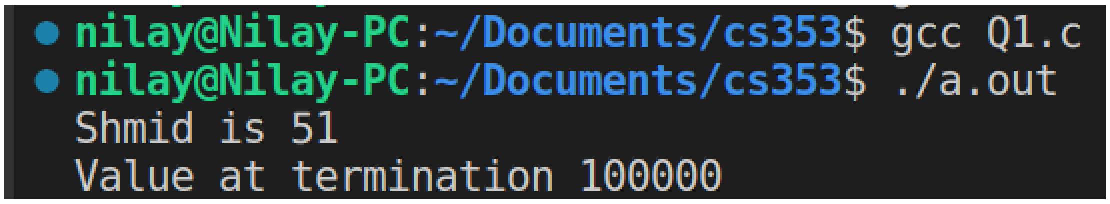
```
#include<sys/ipc.h> //Including required libraries
#include<sys/shm.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/sem.h>
#include<sys/wait.h>
#define SHMSIZE 30
//Driver Code
int main(){
//Initializing Variables
int X=0;
int *s,*shm;
int shmid;
key_t key=5690;
//create the segment
if((shmid=shmget(key, SHMSIZE,0666|IPC_CREAT))<0){
   perror("shmget");
   exit(1);
}
printf("Shmid is %d\n",shmid);
//Attach the segment to our data space
if((shm=(int*)shmat(shmid,NULL,0))==(int*)-1){
   perror("shmat");
   exit(1);
}
//Put the data in Shared memory
*shm=X;
//initializing PIDs and forking child1
pid_t pid1,pid2;
pid1=fork();
if(pid1==0){
   X=*shm;
   for(int i=0;i<100000;i++){
       X++;
       *shm=X;
   }
}else if(pid1>0){
   sleep(1); //solving the race condition by suspending for a sec.
   pid2=fork(); //forking child 2
   if(pid2==0){
       X=*shm;
       for(int i=0;i<100000;i++){
       X--;
       *shm=X;}
   }
   _exit(0);
}
//waiting for child processes to end
int status;
waitpid(pid1,&status,0);
waitpid(pid2,&status,0);
// wait(NULL);
X=*shm;
printf("Value at termination %d \n",X);
//detaching the shared memory
shmdt(s);
shmctl(shmid,IPC_RMID,NULL);
exit(1);
}
```

Input/Output without sleep(); correction:



```
nilay@Nilay-PC:~/Documents/cs353$ gcc Q1.c
nilay@Nilay-PC:~/Documents/cs353$ ./a.out
Shmid is 49
Value at termination 34846
```

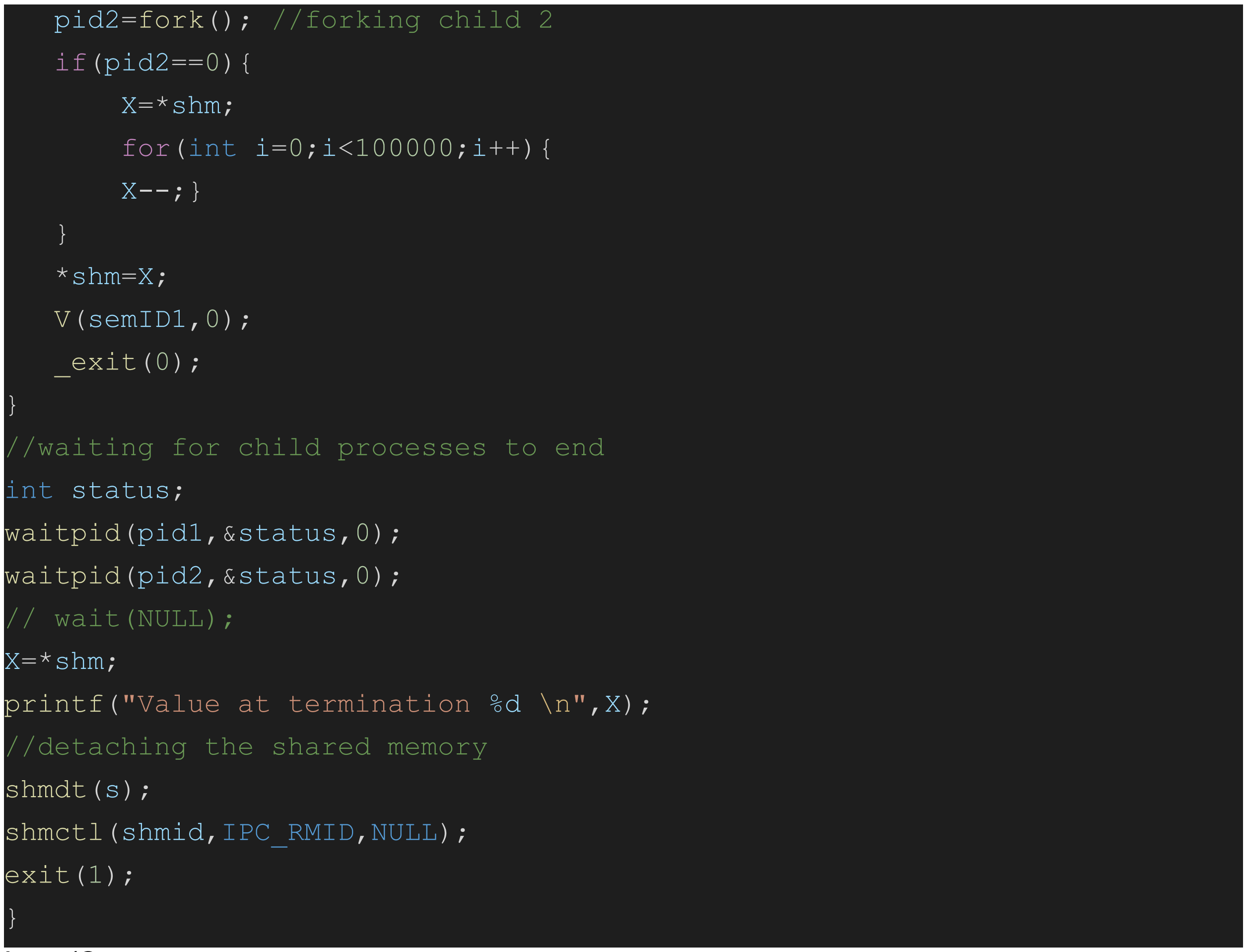
Input/Output with sleep(); correction:



```
nilay@Nilay-PC:~/Documents/cs353$ gcc Q1.c
nilay@Nilay-PC:~/Documents/cs353$ ./a.out
Shmid is 51
Value at termination 100000
```

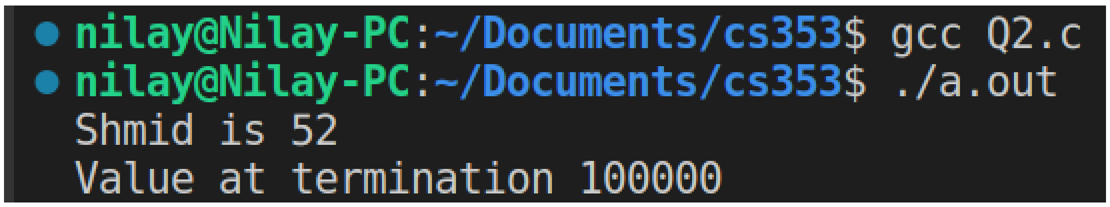
Q2.

```c
#include<sys/ipc.h> //Including required libraries
#include<sys/shm.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/sem.h>
#include<sys/wait.h>
#define SHMSIZE 30

int P(int semId, int semNum) {
   // Operation list of 1 operation, taking resource, no flag
   struct sembuf operationList[1];
   operationList[0].sem_num = semNum;
   operationList[0].sem_op = -1;
   operationList[0].sem_flg = 0;
   return semop(semId, operationList, 1);
}

// Release a resource
int V(int semId, int semNum) {
   // Operation list of 1 operation, releasing resource, no flag
   struct sembuf operationList[1];
   operationList[0].sem_num = semNum;
   operationList[0].sem_op = 1;
   operationList[0].sem_flg = 0;

   return semop(semId, operationList, 1);
}

//Driver Code
int main(){
//Initialising variables
int X=0;
int *s,*shm;
int shmid;
key_t key=5690;
//Variables used for semaphore
int semID1, semID2;
int flag;
char* str;
int count=10;
FILE *fp;
int data=0;
//Initializing Semaphore
semID1=semget(IPC_PRIVATE,1,0666|IPC_CREAT);
semID2=semget(IPC_PRIVATE,1,0666|IPC_CREAT);
semctl(semID1,0,SETVAL,0);
semctl(semID2,0,SETVAL,1);

//create the segment
if((shmid=shmget(key, SHMSIZE,0666|IPC_CREAT))<0){
   perror("shmget");
   exit(1);
}
printf("Shmid is %d\n",shmid);

//Attach the segment to our data space
if((shm=(int*)shmat(shmid,NULL,0))==(int*)-1){
   perror("shmat");
   exit(1);
}

//Put the data in Shared memory
*shm=X;
//initializing PIDs and forking child1
pid_t pid1,pid2;
pid1=fork();

if(pid1==0){
   P(semID1,0); //solving the race condition by using semaphore
   X=*shm;
   for(int i=0;i<100000;i++){
       X++;
   }
   *shm=X;
   V(semID2,0);
}else if(pid1>0){
   P(semID2,0); //solving the race condition by using semaphore
```



```c
    pid2=fork(); //forking child 2
    if(pid2==0){
        X=*shm;
        for(int i=0;i<100000;i++){
        X--;}
    }
    *shm=X;
    V(semID1,0);
    _exit(0);
}
//waiting for child processes to end
int status;
waitpid(pid1,&status,0);
waitpid(pid2,&status,0);
// wait(NULL);
X=*shm;
printf("Value at termination %d \n",X);
//detaching the shared memory
shmdt(s);
shmctl(shmid,IPC_RMID,NULL);
exit(1);
}
```

Input/Output:



```
● nilay@Nilay-PC:~/Documents/cs353$ gcc Q2.c
● nilay@Nilay-PC:~/Documents/cs353$ ./a.out
  Shmid is 52
  Value at termination 100000
```