/*

NAME : PAARTH SINGH

SECTION : iOS

UNIVERSITY ROLL NO : 2023496

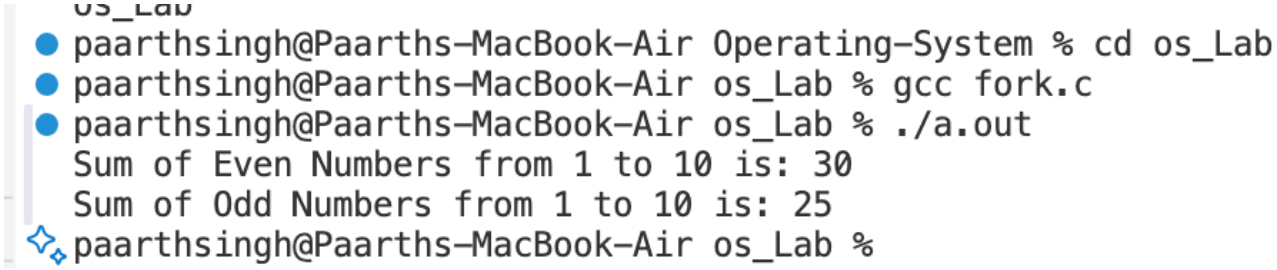
*/

Practical 1. Write a C Program to implement fork() system call.

Code :-

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main(){
   int sume = 0 , sumo = 0;
   int pid ;
   pid = fork();
   if (pid < 0){
      printf("Fork Failed\n");
      exit(1);
   }
   else if (pid == 0){
      for (int i = 1; i <= 10; i+=2){
         sumo += i;
      }
      printf("Sum of Odd Numbers from 1 to 10 is: %d\n", sumo);
   }
   else{
      for (int i = 0; i <= 10; i+=2){
         sume += i;
      }
      printf("Sum of Even Numbers from 1 to 10 is: %d\n", sume);
   }
}
```

OUTPUT :-


os_Lab
paarthsingh@Paarths-MacBook-Air Operating-System % cd os_Lab
paarthsingh@Paarths-MacBook-Air os_Lab % gcc fork.c
paarthsingh@Paarths-MacBook-Air os_Lab % ./a.out
Sum of Even Numbers from 1 to 10 is: 30
Sum of Odd Numbers from 1 to 10 is: 25
paarthsingh@Paarths-MacBook-Air os_Lab %

/*

NAME : PAARTH SINGH

SECTION : iOS

UNIVERSITY ROLL NO : 2023496

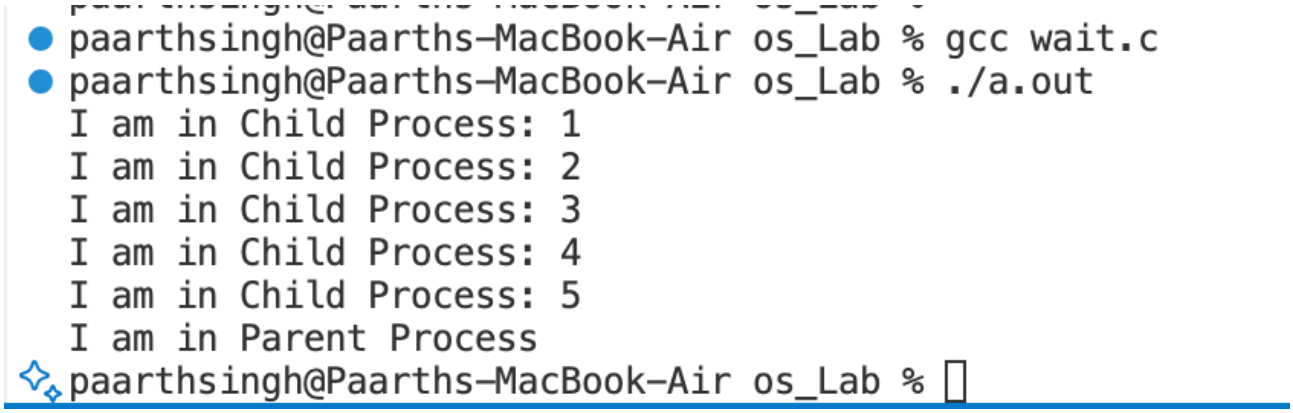
*/

Practical 2. Write a C Program to implement wait() system call.

Code :-

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
int main(){
    int pid = fork();
    if (pid > 0){
        wait(NULL);
        printf("I am in Parent Process\n");
    }
    else if (pid == 0){
        for (int i = 1; i <= 5; i++){
            printf("I am in Child Process: %d\n", i);
        }
    }
    else{
        printf("Fork Failed\n");
    }
    return 0;
}
```

OUTPUT:-


paarthsingh@Paarths-MacBook-Air os_Lab % gcc wait.c
paarthsingh@Paarths-MacBook-Air os_Lab % ./a.out
I am in Child Process: 1
I am in Child Process: 2
I am in Child Process: 3
I am in Child Process: 4
I am in Child Process: 5
I am in Parent Process
paarthsingh@Paarths-MacBook-Air os_Lab %

/*

NAME : PAARTH SINGH

SECTION : iOS

UNIVERSITY ROLL NO : 2023496

*/

Practical 3. Write a C Program to implement Zombie Process.

Code :-

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main(){
   int pid = fork();
   if (pid > 0){
      printf("I am in Parent Process\n");
      sleep(100);
      printf("Still Alive my ID is %d\n", getpid());
   }
   else if (pid == 0){
      printf("I am in Child Process and my ID is %d and my Parents ID is %d\n", getpid(),
getppid());
   }
   else{
      printf("Fork Failed\n");
   }
   return 0;
}
```

OUTPUT:-

```
paarthsingh@Paarths-MacBook-Air os_Lab % gcc zombie.c
paarthsingh@Paarths-MacBook-Air os_Lab % ./a.out
I am in Parent Process
I am in Child Process and my ID is 5016 and my Parents ID is 5015
Still Alive my ID is 5015
paarthsingh@Paarths-MacBook-Air os_Lab %
```

/*

NAME : PAARTH SINGH

SECTION : iOS

UNIVERSITY ROLL NO : 2023496

*/

Practical 4. Write a C Program to implement Orphan Process.

Code :-

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main(){
   int pid = fork();
   if (pid > 0){
      printf("I am in Parent Process with Child my ID is %d\n", getpid());
      exit(0);
   }
   else if (pid == 0){
      printf("I am in Child Process and my Parents ID is %d\n", getppid());
      sleep(5);
      printf("After 5 seconds my Parents ID is %d\n", getppid());
   }
   else{
      printf("Fork Failed\n");
   }
   return 0;
}
```

OUTPUT:-

```
paarthsingh@Paarths-MacBook-Air os_Lab % gcc orphan.c
paarthsingh@Paarths-MacBook-Air os_Lab % ./a.out
I am in Parent Process with Child my ID is 5246
I am in Child Process and my Parents ID is 5246
paarthsingh@Paarths-MacBook-Air os_Lab % After 5 seconds my Parents ID is 1

paarthsingh@Paarths-MacBook-Air os_Lab %
```

/*

NAME : PAARTH SINGH

SECTION : iOS

UNIVERSITY ROLL NO : 2023496

*/

Practical 5. Write a C Program to implement FCFS Scheduling Algorithm.

Code :-

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   int n;
   printf("Enter the number of processes: ");
   scanf("%d", &n);
   int PID[n], AT[n], BT[n], WT[n], TAT[n], ET[n];
   float avgWT = 0, avgTAT = 0;
   for (int i = 0; i < n; i++) {
      printf("Enter process ID for process %d: ", i + 1);
      scanf("%d", &PID[i]);
      printf("Enter arrival time for process %d: ", i + 1);
      scanf("%d", &AT[i]);
      printf("Enter burst time for process %d: ", i + 1);
      scanf("%d", &BT[i]);
   }
   for (int i = 0; i < n - 1; i++) {
      for (int j = 0; j < n - i - 1; j++) {
         if (AT[j] > AT[j + 1]) {
            int temp = AT[j];
            AT[j] = AT[j + 1];
            AT[j + 1] = temp;

            temp = BT[j];
            BT[j] = BT[j + 1];
            BT[j + 1] = temp;

            temp = PID[j];
            PID[j] = PID[j + 1];
            PID[j + 1] = temp;
         }
      }
   }
   ET[0] = AT[0] + BT[0];
   for (int i = 1; i < n; i++) {
```

```
        if (ET[i - 1] < AT[i]) {
            ET[i] = AT[i] + BT[i];
        } else {
            ET[i] = ET[i - 1] + BT[i];
        }
    }
    for (int i = 0; i < n; i++) {
        TAT[i] = ET[i] - AT[i];
        WT[i] = TAT[i] - BT[i];
        avgWT += WT[i];
        avgTAT += TAT[i];
    }
    avgWT /= n;
    avgTAT /= n;

printf("\nPROCESS_ID\tARRIVAL_TIME\tBURST_TIME\tWAITING_TIME\tTURN_AROUND
_TIME\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", PID[i], AT[i], BT[i], WT[i], TAT[i]);
    }
    printf("\nAverage Waiting Time: %.2f", avgWT);
    printf("\nAverage Turnaround Time: %.2f\n", avgTAT);

    return 0;
}
```

OUTPUT:-

```
paarthsingh@Paarths-MacBook-Air os_Lab % gcc fcfs.c
paarthsingh@Paarths-MacBook-Air os_Lab % ./a.out
Enter the number of processes: 4
Enter process ID for process 1: 1
Enter arrival time for process 1: 2
Enter burst time for process 1: 10
Enter process ID for process 2: 2
Enter arrival time for process 2: 3
Enter burst time for process 2: 8
Enter process ID for process 3: 3
Enter arrival time for process 3: 1
Enter burst time for process 3: 7
Enter process ID for process 4: 4
Enter arrival time for process 4: 3
Enter burst time for process 4: 5

PROCESS_ID      ARRIVAL_TIME    BURST_TIME      WAITING_TIME    TURN_AROUND_TIME
3               1               7               0               7
1               2               10              6               16
2               3               8               15              23
4               3               5               23              28

Average Waiting Time: 11.00
Average Turnaround Time: 18.50
paarthsingh@Paarths-MacBook-Air os_Lab %
```

/*

NAME : PAARTH SINGH

SECTION : iOS

UNIVERSITY ROLL NO : 2023496

*/

Practical 6. Write a C Program to implement Shortest Job First (SJF) Scheduling Algorithm.

Code :-

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

int main() {
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);

    int PID[n], BT[n], WT[n], TAT[n], ET[n];
    float avgWT = 0, avgTAT = 0;

    for (int i = 0; i < n; i++) {
        printf("Enter process ID for process %d: ", i + 1);
        scanf("%d", &PID[i]);
        printf("Enter burst time for process %d: ", i + 1);
        scanf("%d", &BT[i]);
    }

    // Sort based on Burst Time (SJF logic)
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (BT[j] > BT[j + 1]) {
                int temp = BT[j];
                BT[j] = BT[j + 1];
                BT[j + 1] = temp;

                temp = PID[j];
                PID[j] = PID[j + 1];
                PID[j + 1] = temp;
            }
        }
    }
```

```
    // Since no arrival time, CPU starts at 0
    ET[0] = BT[0];
    for (int i = 1; i < n; i++) {
        ET[i] = ET[i - 1] + BT[i];
    }

    for (int i = 0; i < n; i++) {
        TAT[i] = ET[i];
        WT[i] = TAT[i] - BT[i];
        avgWT += WT[i];
        avgTAT += TAT[i];
    }

    avgWT /= n;
    avgTAT /= n;

printf("\nPROCESS_ID\tBURST_TIME\tWAITING_TIME\tTURN_AROUND_TIME\n");
for (int i = 0; i < n; i++) {
        printf("%d\t\t%d\t\t%d\t\t%d\n", PID[i], BT[i], WT[i], TAT[i]);
    }

    printf("\nAverage Waiting Time: %.2f", avgWT);
    printf("\nAverage Turnaround Time: %.2f\n", avgTAT);

    return 0;
}
```

OUTPUT:-

```
paarthsingh@Paarths-MacBook-Air os_Lab % gcc sjf.c
paarthsingh@Paarths-MacBook-Air os_Lab % ./a.out
Enter the number of processes: 4
Enter process ID for process 1: 1
Enter burst time for process 1: 6
Enter process ID for process 2: 2
Enter burst time for process 2: 8
Enter process ID for process 3: 3
Enter burst time for process 3: 3
Enter process ID for process 4: 4
Enter burst time for process 4: 7

PROCESS_ID      BURST_TIME      WAITING_TIME    TURN_AROUND_TIME
3               3               0               3
1               6               3               9
4               7               9               16
2               8               16              24

Average Waiting Time: 7.00
Average Turnaround Time: 13.00
```

/*

NAME : PAARTH SINGH

SECTION : iOS

UNIVERSITY ROLL NO : 2023496

*/

Practical 7. Write a C Program to implement Priority Scheduling Algorithm.

Code :-

```
#include<stdio.h>
#include<stdlib.h>
int main(){
   int n ;
   printf("Enter the Number of Processed :");
   scanf("%d", &n);
   int PID[n] , BT[n], priority[n] , ET[n] , WT[n], TAT[n] ;
   for (int i = 0 ; i<n ;i++){
      printf("Enter the ProcessID for %d process ", i+1);
      scanf("%d",&PID[i]);
      printf("Enter burst time for process %d: ", i + 1);
      scanf("%d", &BT[i]);
      printf("Enter priority for process %d (lower value means higher priority): ", i + 1);
      scanf("%d", &priority[i]);
   }
      for (int i = 0; i < n - 1; i++) {
      for (int j = 0; j < n - i - 1; j++) {
         if (priority[j] < priority[j + 1] || (priority[j] == priority[j + 1] && PID[j] > PID[j + 1])) {
            int temp = priority[j];
            priority[j] = priority[j + 1];
            priority[j + 1] = temp;

            temp = BT[j];
            BT[j] = BT[j + 1];
            BT[j + 1] = temp;

            temp = PID[j];
            PID[j] = PID[j + 1];
            PID[j + 1] = temp;
         }
      }
   }
   ET[0] = BT[0];
   for (int i = 1; i < n; i++) {
      ET[i] = ET[i - 1] + BT[i];
```

```
    }
    float avgWT = 0, avgTAT = 0;
    for (int i = 0; i < n; i++) {
        TAT[i] = ET[i];
        WT[i] = TAT[i] - BT[i];
        avgWT += WT[i];
        avgTAT += TAT[i];
    }

    printf("PROCESS_ID\t, PRIORITY\t, BURST_TIME\t, WAITING_TIME\t,
TURN_AROUND_TIME\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t\t, %d\t\t, %d\t\t, %d\t\t, %d\n", PID[i], priority[i], BT[i], WT[i], TAT[i]);
    }
    printf("\nAverage Waiting Time: %f", avgWT/n);
    printf("\nAverage Turnaround Time: %f\n",avgTAT/n);
    return 0;
}
```

OUTPUT:-

```
paarthsingh@Paarths-MacBook-Air os_Lab % gcc priority_without_AT.c
paarthsingh@Paarths-MacBook-Air os_Lab % ./a.out
Enter the Number of Processed :4
Enter the ProcessID for 1 process 1
Enter burst time for process 1: 7
Enter priority for process 1 (lower value means higher priority): 10
Enter the ProcessID for 2 process 2
Enter burst time for process 2: 6
Enter priority for process 2 (lower value means higher priority): 3
Enter the ProcessID for 3 process 3
Enter burst time for process 3: 8
Enter priority for process 3 (lower value means higher priority): 6
Enter the ProcessID for 4 process 4
Enter burst time for process 4: 9
Enter priority for process 4 (lower value means higher priority): 7
PROCESS_ID      , PRIORITY      , BURST_TIME    , WAITING_TIME  , TURN_AROUND_TIME
1               , 10            , 7             , 0             , 7
4               , 7             , 9             , 7             , 16
3               , 6             , 8             , 16            , 24
2               , 3             , 6             , 24            , 30

Average Waiting Time: 11.750000
Average Turnaround Time: 19.250000
paarthsingh@Paarths-MacBook-Air os_Lab %
```

/*

NAME : PAARTH SINGH

SECTION : iOS

UNIVERSITY ROLL NO : 2023496

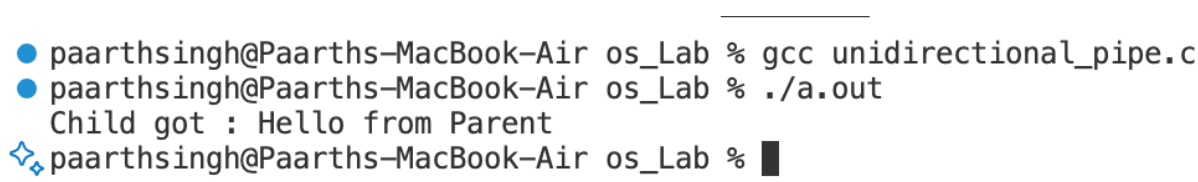
*/

Practical 8. Write a C Program to implement pipe() system call.

Code :-

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main(){
   char buffer[50];
   int fd[2];
   char message[] = "Hello from Parent";
   pipe(fd);
   if (fork()==0){
      read(fd[0],buffer,sizeof(buffer));
      printf("Child got : %s\n",buffer);
   }
   else{
      write(fd[1],message,sizeof(message));
   }
   return 0;
}
```

OUTPUT:-


paarthsingh@Paarths-MacBook-Air os_Lab % gcc unidirectional_pipe.c
paarthsingh@Paarths-MacBook-Air os_Lab % ./a.out
Child got : Hello from Parent
paarthsingh@Paarths-MacBook-Air os_Lab %

/*

NAME : PAARTH SINGH

SECTION : iOS

UNIVERSITY ROLL NO : 2023496

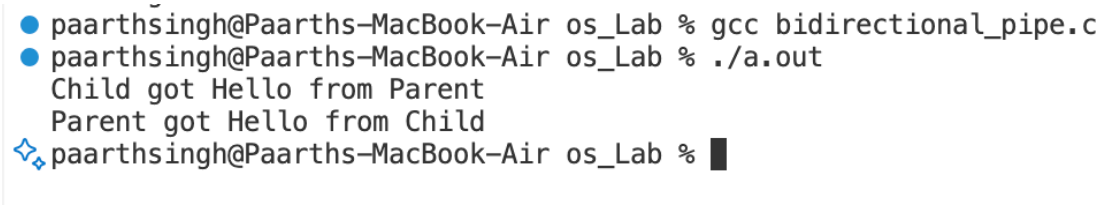
*/

Practical 9. Write a C Program to implement bidirectional pipe() system call.

Code :-

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main(){
   char buffer[50];
   int fd1[2], fd2[2];
   char messageChild[] = "Hello from Child";
   char messageParent[] = "Hello from Parent";
   pipe(fd1);
   pipe(fd2);
   if (fork()==0){
      write(fd2[1],messageChild,sizeof(messageChild));
      read(fd1[0],buffer,sizeof(buffer));
      printf("Child got %s\n",buffer);
   }else{
      write(fd1[1],messageParent,sizeof(messageParent));
      read(fd2[0],buffer,sizeof(buffer));
      printf("Parent got %s\n",buffer);
   }
   return 0;
}
```

OUTPUT:-


paarthsingh@Paarths-MacBook-Air os_Lab % gcc bidirectional_pipe.c
paarthsingh@Paarths-MacBook-Air os_Lab % ./a.out
Child got Hello from Parent
Parent got Hello from Child
paarthsingh@Paarths-MacBook-Air os_Lab %

/*

NAME : PAARTH SINGH

SECTION : iOS

UNIVERSITY ROLL NO : 2023496

*/

Practical 10. Write a C Program to implement FIFO (Named pipe).

Code :-

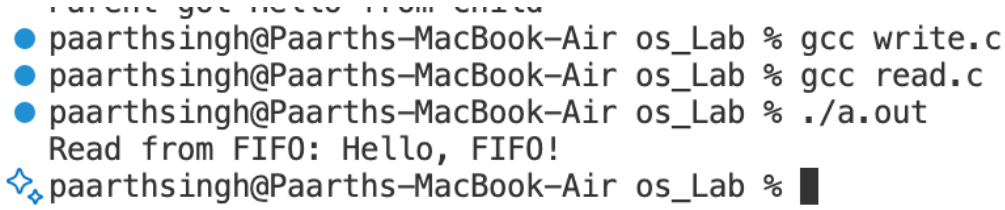
Writer File

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <unistd.h>

int main() {
    int fd[2];
    char* message = "Hello, FIFO!";
    char * FIFO_FILE = "fifo.txt";
    mkfifo(FIFO_FILE, 0666);
    fd[1] = open(FIFO_FILE, O_WRONLY);
    write(fd[1], message, strlen(message) + 1);
    printf("Wrote to FIFO: %s\n", message);
    close(fd[1]);

    return 0;
}
```

Reader File

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <unistd.h>
int main() {
    int fd[2];
    char buffer[100];
    char * FIFO_FILE = "fifo.txt";
    fd[0] = open(FIFO_FILE, O_RDONLY);
    read(fd[0], buffer, 100);
    printf("Read from FIFO: %s\n", buffer);
    close(fd[0]);
    return 0;
}
```

OUTPUT:-


paarthsingh@Paarths-MacBook-Air os_Lab % gcc write.c
paarthsingh@Paarths-MacBook-Air os_Lab % gcc read.c
paarthsingh@Paarths-MacBook-Air os_Lab % ./a.out
Read from FIFO: Hello, FIFO!
paarthsingh@Paarths-MacBook-Air os_Lab %

/*

NAME : PAARTH SINGH

SECTION : iOS

UNIVERSITY ROLL NO : 2023496

*/

Practical 11. Write a C Program to implement Message Queue.

Code :-

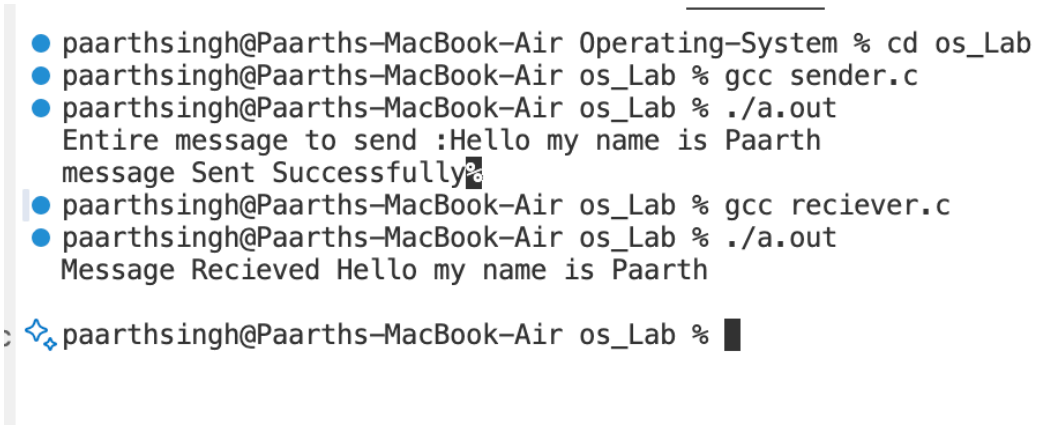
Sender File

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/ipc.h>
#include<sys/msg.h>
struct message{
    long mtype;
    char mtext[100];
}message;
int main (){
    key_t key;
    int msgid;
    key = ftok("progfile",65);
    msgid = msgget(key,0666 | IPC_CREAT);
    printf("Entire message to send :");
    fgets(message.mtext,sizeof(message.mtext),stdin);
    message.mtype = 1;
    msgsnd(msgid, &message, sizeof(message),0);
    printf("message Sent Successfully");
    return 0;
}
```

Receiver File

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/ipc.h>
#include<sys/msg.h>
struct message{
    long mtype;
    char mtext[100];
}message;
int main(){
    key_t key;
    int msgid;
    key = ftok("progfile",65);
    msgid = msgget(key,0666 | IPC_CREAT);
    msgrcv(msgid , &message , sizeof(message),1,0);
    printf("Message Recieved %s\n",message.mtext);
    msgctl(msgid,IPC_RMID,NULL);
    return 0 ;
}
```

OUTPUT:-


paarthsingh@Paarths-MacBook-Air Operating-System % cd os_Lab
paarthsingh@Paarths-MacBook-Air os_Lab % gcc sender.c
paarthsingh@Paarths-MacBook-Air os_Lab % ./a.out
Entire message to send :Hello my name is Paarth
message Sent Successfully
paarthsingh@Paarths-MacBook-Air os_Lab % gcc reciever.c
paarthsingh@Paarths-MacBook-Air os_Lab % ./a.out
Message Recieved Hello my name is Paarth
paarthsingh@Paarths-MacBook-Air os_Lab %

/*

NAME : PAARTH SINGH

SECTION : iOS

UNIVERSITY ROLL NO : 2023496

*/

Practical 12. Write a C Program to implement Shared Memory.

Code :-

Writer File

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/msg.h>
#include<sys/ipc.h>
#include <sys/shm.h>
#include <string.h>
int main(){
    key_t key = ftok("shmfile",65);
    int shmid = shmget(key,1024,0666 | IPC_CREAT);
    char *str = (char*)shmat(shmid,(void*)0,0);
    strcpy(str,"Hello Shared Memory");
    shmdt(str);
    return 0;
}
```

Reader File

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/msg.h>
#include<sys/ipc.h>
#include <sys/shm.h>
#include <string.h>
int main(){
    key_t key = ftok("shmfile",65);
    int shmid = shmget(key,1024,0666);
    char *str = (char*)shmat(shmid,(void*)0,0);
    printf("Message = %s\n",str);
    shmdt(str);
    shmctl(shmid, IPC_RMID, NULL);
    return 0;
}
```

OUTPUT:-

```
paarthsingh@Paarths-MacBook-Air os_Lab % gcc shmwriter.c
paarthsingh@Paarths-MacBook-Air os_Lab % ./a.out
paarthsingh@Paarths-MacBook-Air os_Lab % gcc shmreader.c
paarthsingh@Paarths-MacBook-Air os_Lab % ./a.out
Message = Hello Shared Memory
paarthsingh@Paarths-MacBook-Air os_Lab %
```

/*

NAME : PAARTH SINGH

SECTION : iOS

UNIVERSITY ROLL NO : 2023496

*/

Practical 13. Write a C Program to implement FIFO Page Replacement Algorithm.

Code :-

```
#include <stdio.h>
int main() {
   int n, f, pages[50], frames[10], count = 0, pageFaults = 0, i, j, k = 0, flag;
   printf("Enter number of pages: ");
   scanf("%d", &n);
   printf("Enter the page reference string:\n");
   for (i = 0; i < n; i++)
      scanf("%d", &pages[i]);

   printf("Enter number of frames: ");
   scanf("%d", &f);

   for (i = 0; i < f; i++)
      frames[i] = -1;

   printf("\nPage\tFrames\n");
   for (i = 0; i < n; i++) {
      flag = 0;
      for (j = 0; j < f; j++) {
         if (frames[j] == pages[i]) {
            flag = 1;
            break;
         }
      }
      if (flag == 0) {
         frames[k] = pages[i];
         k = (k + 1) % f;
         pageFaults++;
      }
      printf("%d\t", pages[i]);
      for (j = 0; j < f; j++) {
         if (frames[j] != -1)
            printf("%d ", frames[j]);
         else
            printf("- ");
```

```
        }
        printf("\n");
    }
    printf("\nTotal Page Faults = %d\n", pageFaults);
    return 0;
}
```

OUTPUT:-

```
paarthsingh@Paarths-MacBook-Air os_Lab % gcc FIFO_Page_Replacement.c
paarthsingh@Paarths-MacBook-Air os_Lab % ./a.out
Enter number of pages: 12
Enter the page reference string:
1 2 3 4 1 2 5 1 2 3 4 5
Enter number of frames: 3

Page    Frames
1       1 - -
2       1 2 -
3       1 2 3
4       4 2 3
1       4 1 3
2       4 1 2
5       5 1 2
1       5 1 2
2       5 1 2
3       5 3 2
4       5 3 4
5       5 3 4

Total Page Faults = 9
paarthsingh@Paarths-MacBook-Air os_Lab %
```

/*

NAME : PAARTH SINGH

SECTION : iOS

UNIVERSITY ROLL NO : 2023496

*/

Practical 14. Write a C Program to implement Least Recently Used Page Replacement Algorithm.

Code :-

```
#include <stdio.h>
int main() {
   int n, f, pages[50], frames[10], recent[10], pageFaults = 0, i, j, k, pos, min;
   printf("Enter number of pages: ");
   scanf("%d", &n);
   printf("Enter the page reference string:\n");
   for (i = 0; i < n; i++)
      scanf("%d", &pages[i]);

   printf("Enter number of frames: ");
   scanf("%d", &f);
   for (i = 0; i < f; i++) {
      frames[i] = -1;
      recent[i] = 0;
   }
   printf("\nPage\tFrames\n");
   for (i = 0; i < n; i++) {
      int flag = 0;
      for (j = 0; j < f; j++) {
         if (frames[j] == pages[i]) {
            flag = 1;
            recent[j] = i + 1;
            break;
         }
      }
      if (flag == 0) {
         if (i < f) {
            frames[i] = pages[i];
            recent[i] = i + 1;
         } else {
            min = recent[0];
            pos = 0;
            for (k = 1; k < f; k++) {
               if (recent[k] < min) {
                  min = recent[k];
```

```
                pos = k;
            }
        }
        frames[pos] = pages[i];
        recent[pos] = i + 1;
      }
      pageFaults++;
    }
    printf("%d\t", pages[i]);
    for (j = 0; j < f; j++) {
      if (frames[j] != -1)
        printf("%d ", frames[j]);
      else
        printf("- ");
    }
    printf("\n");
  }
  printf("\nTotal Page Faults = %d\n", pageFaults);
  return 0;
}
```

OUTPUT:-

```
paarthsingh@Paarths-MacBook-Air os_Lab % gcc LRU_Page_Replacement.c
paarthsingh@Paarths-MacBook-Air os_Lab % ./a.out
Enter number of pages: 12
Enter the page reference string:
1 2 3 4 1 2 5 1 2 3 4 5
Enter number of frames: 3

Page    Frames
1       1 - -
2       1 2 -
3       1 2 3
4       4 2 3
1       4 1 3
2       4 1 2
5       5 1 2
1       5 1 2
2       5 1 2
3       3 1 2
4       3 4 2
5       3 4 5

Total Page Faults = 10
paarthsingh@Paarths-MacBook-Air os_Lab %
```