Assignment 2

Name: Atharva Agrawal
Roll No: 33303
Batch: L11

Program:

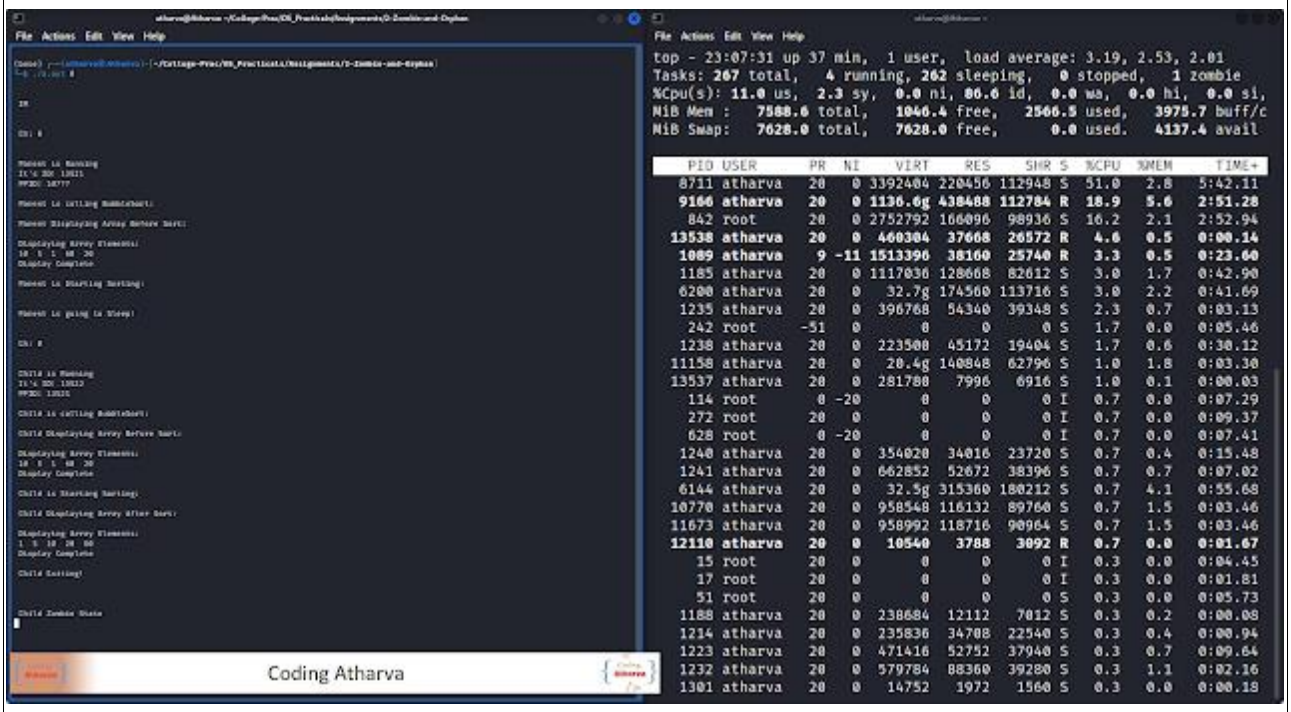
1) Parent Child Sort

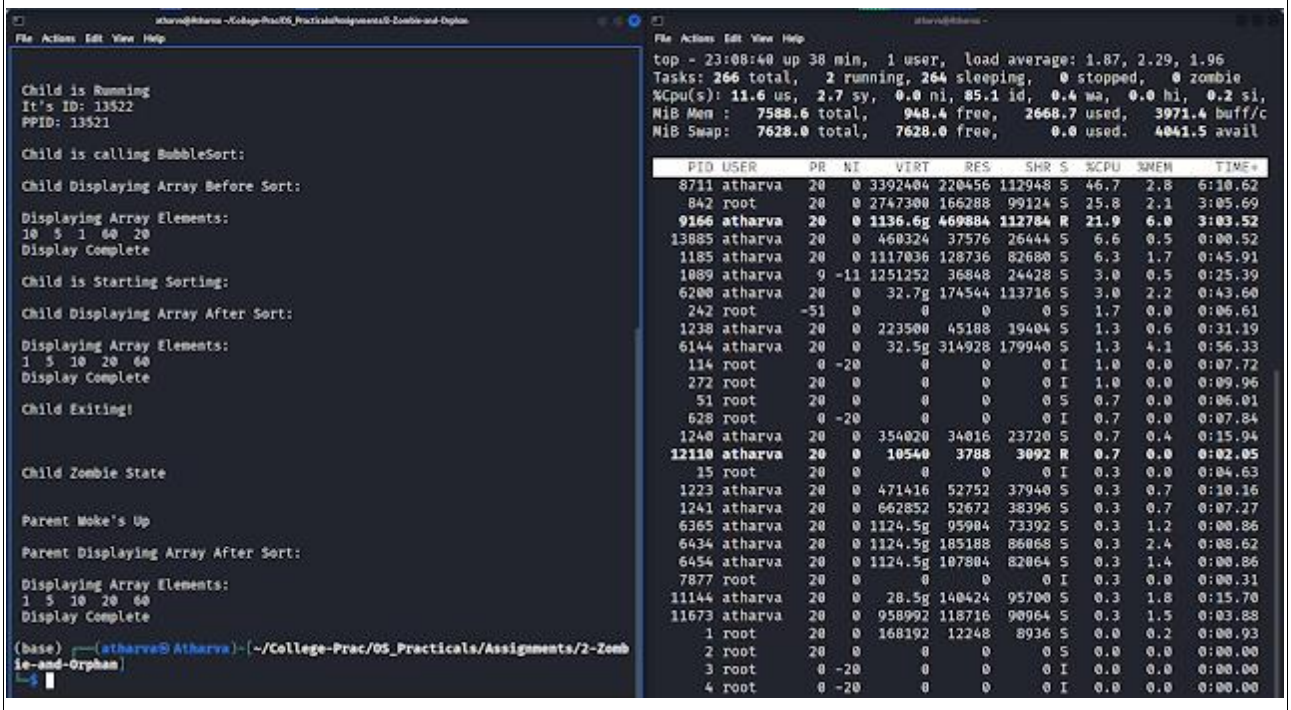
```
#include<stdio.h>

void bubbleSort(int arr[],int n)
{
   for(int i=0;i<n;i++)
   {
      for(int j=0;j<n-1;j++)
      {
         if(arr[j] > arr[j+1])
         {
            int temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
         }
      }
   }
}

void displayArr(int arr[],int n)
{
   printf("\n Displaying Array Elements:\n");
   for(int i=0;i<n;i++)
   {
      printf(" %d ",arr[i]);
   }
   printf("\n Display Complete \n");
}

// 0 - Zombie State
// 1 - Orphan State
void main(int argc, char *argv[])
{
   printf("\n \n IM \n");
   int cpid = fork();

   int arr[] = {10,5,1,60,20};
   int n = 5;
   int ch = atoi(argv[1]);

   printf("\n\n Ch: %d \n",ch);
   if(cpid > 0)
   {
      printf("\n\n Parent is Running\n It's ID: %d \n PPID: %d \n",getpid(),getppid());
      printf("\n Parent is calling BubbleSort:\n");
      printf("\n Parent Displaying Array Before Sort:\n");
      displayArr(arr,n);
      printf("\n Parent is Starting Sorting:\n");
```

```
    bubbleSort(arr,n);
    printf("\n\n Parent is going to Sleep!\n");

    if(ch == 0)
    {
      //printf("\n\n Parent Zombie State \n");
      sleep(10);
      wait(NULL);
    }
    else
    {
      //printf("\n\n Parent Orphan State \n");
      sleep(1);
    }

    printf("\n\n Parent Woke's Up");
    printf("\n\n Parent Displaying Array After Sort:\n");
    displayArr(arr,n);
    if(ch == 1)
    {
      printf("\n Parent is Exiting!\n");
    }
  }
  else if(cpid == 0)
  {
    printf("\n\n Child is Running \n It's ID: %d \n PPID: %d \n",getpid(),getppid());
    printf("\n Child is calling BubbleSort:\n");
    printf("\n Child Displaying Array Before Sort:\n");
    displayArr(arr,n);
    printf("\n Child is Starting Sorting:\n");
    bubbleSort(arr,n);
    if(ch == 1)
    {
      printf("\n Child is goint to Sleep \n");
      sleep(10);
      printf("\n\n Child Orphan State \n It's ID: %d \n PPID: %d \n",getpid(),getppid());
    }

    printf("\n Child Displaying Array After Sort:\n");
    displayArr(arr,n);
    printf("\n Child Exiting!\n\n");

    if(ch == 0)
    {
      printf("\n\n Child Zombie State \n");
    }
  }
}
```

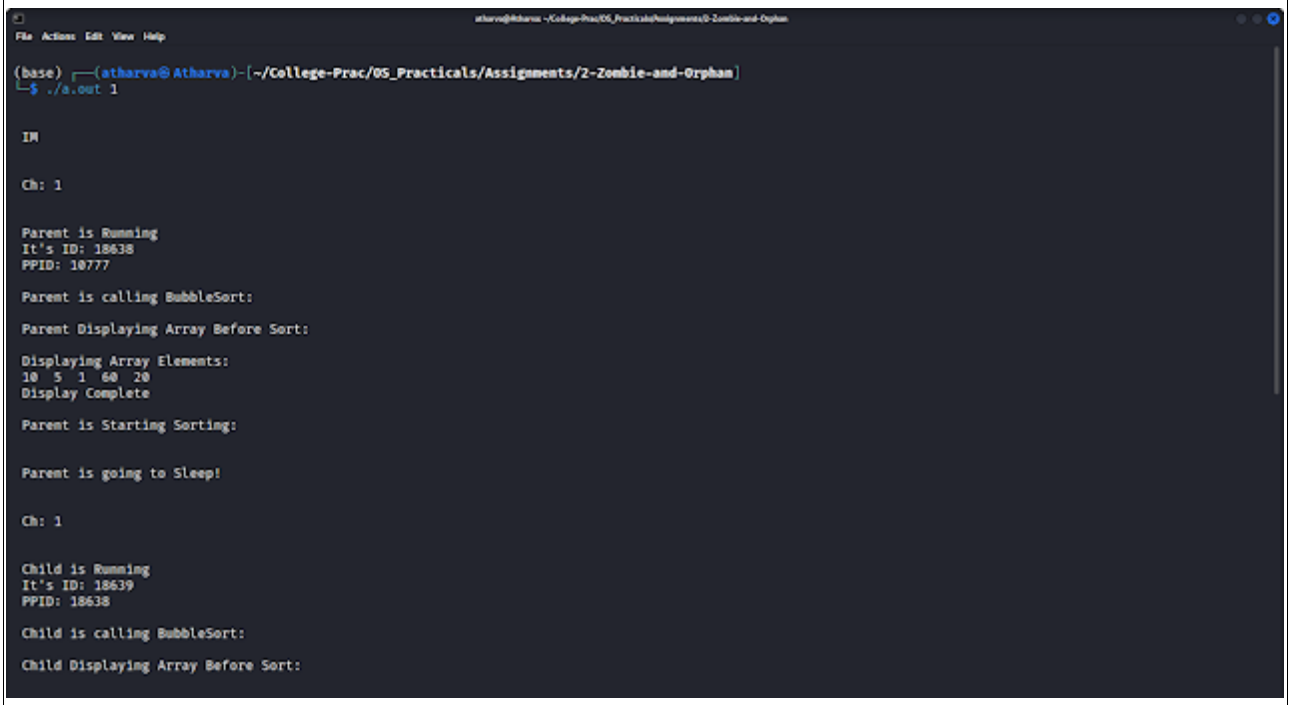
Output:

Command Line Argument 0 Zombie State:





Command Line Argument 1 Orphan State:



2) Parent Process Sorting, Child Process Displaying in Reverse Order via EXCEV

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

void bubbleSort(int arr[],int n)
{
   printf("\n Sorting Started \n");

   for(int i=0;i<n;i++)
   {
      for(int j=0;j<n-1;j++)
      {
         if(arr[j] > arr[j+1])
         {
            int temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
         }
      }
   }

   printf("\n\n Sorting Completed! \n");
}

// To Convert Integer to Char*
void tostring(char str[], int num)
{
   int i, rem, len = 0, n;

   n = num;
   while (n != 0)
   {
      len++;
      n /= 10;
   }
   for (i = 0; i < len; i++)
   {
      rem = num % 10;
      num = num / 10;
      str[len - (i + 1)] = rem + '0';
   }
   str[len] = '\0';
}

void main(int argc, char *argv[])
{
   printf("\n This is the main process: ");
   printf("\n Process Id: %d",getpid());
      printf("\n Parent Id: %d",getppid());


   int arr[] = {10,5,1,60,20};
   int n = 5;
   printf("\n\n Sorting Array using Bubble Sort:");
   bubbleSort(arr,n);
```

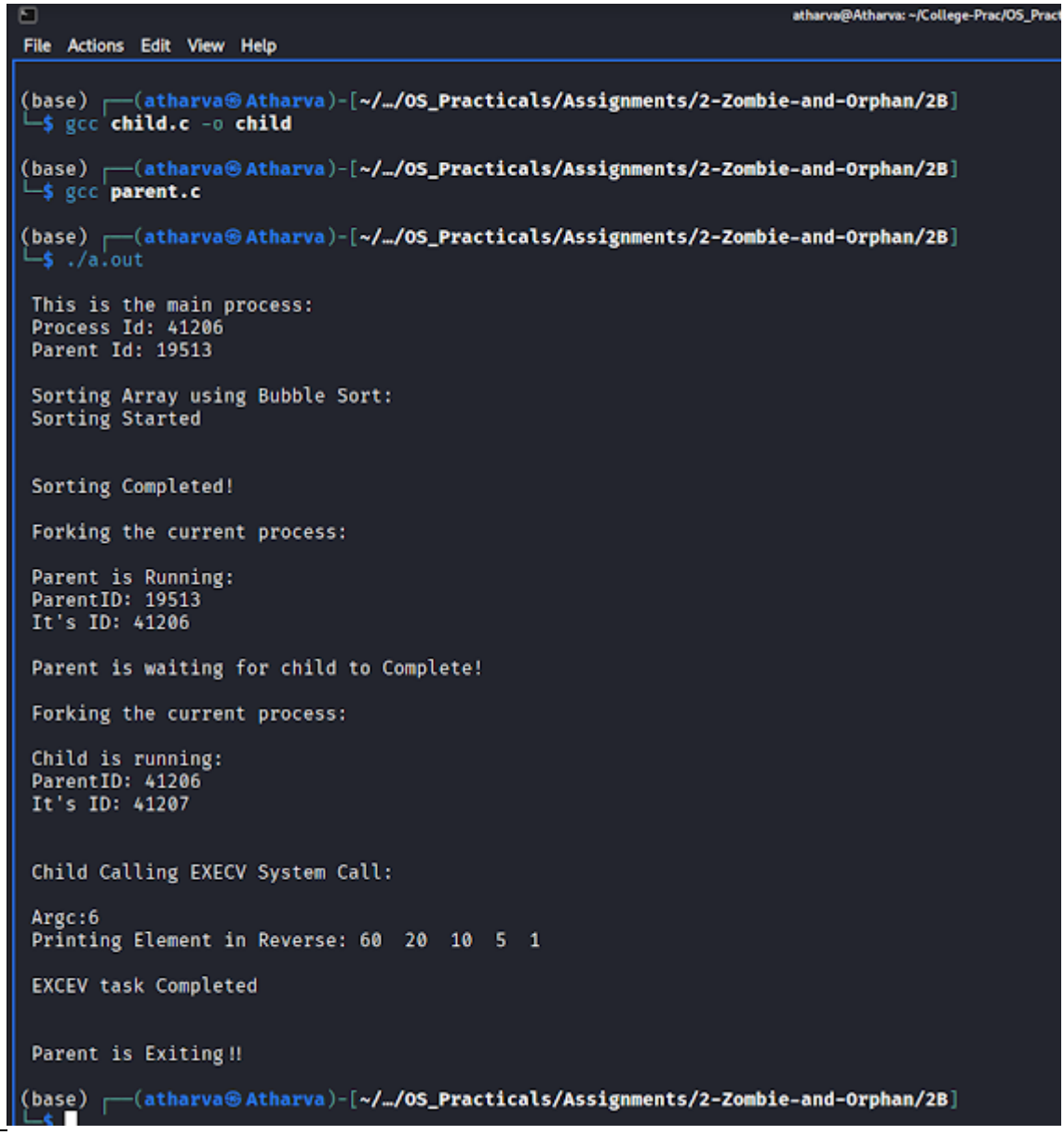
```
  printf("\n Forking the current process:");
  pid_t cpid = fork();
  //The pid_t data type is a signed integer type which is capable of representing a process ID.


  if(cpid > 0)
  {
    printf("\n\n Parent is Running:\n ParentID: %d \n It's ID: %d \n",getppid(),getpid());

    printf("\n Parent is waiting for child to Complete! \n\n");

    wait(NULL);

    printf("\n\n Parent is Exiting!!\n");
  }
  else if(cpid == 0)
  {
    printf("\n\n Child is running:\n ParentID: %d \n It's ID: %d \n",getppid(),getpid());

    char *arrChar[n+1];

    // Creating Ascii Character Array to Pass
    // as command line Argument
    arrChar[0] = (char *) "child";   // Arg 0 = name of executable file
    for(int i=0;i<n;i++)
    {
      char *string = malloc (sizeof(char) * (20));
      tostring(string,arr[i]);
      arrChar[i+1] = string;
    }
    arrChar[n+1] = NULL;

    printf("\n\n Child Calling EXECV System Call:\n");
    execv("./child",arrChar);

    printf("\n\n Child EXECV Call Complete\n");
    printf("\n\n Child Execution Complete \n");
  }
  else if(cpid < 0)
  {
    printf("Error");
  }
}
```

Program Child.c:

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

void main(int argc, char *argv[])
{
  /* argv[0] is the program name */

  int *data = (int *) malloc((argc) * sizeof(int));

  printf("\n Argc:%d",argc);
  for(int i = 0;i < argc;i++)
```

```
    {
        data[i] = atoi(argv[i]);
    }

    // Printing Element in Reverse
    printf("\n Printing Element in Reverse:");
    for(int i = argc-1; i>0;i--)
    {
        printf(" %d ",data[i]);
    }

    printf("\n\n EXCEV task Completed \n");
}
```

Output:



Assignment 3

Name: Atharva Agrawal
Roll No: 33303
Batch: L11

Program : Shortest Job First (Non-Preemptive)

```
#include <iostream>
#include <set>
using namespace std;

int main()
{
    freopen("input.txt", "r", stdin);

    int n, temp, i, j;
    int total_time_cpu = 0;
    float total_waiting_time = 0, total_turnaround_time = 0, cpu_idle_time = 0;

    cout << "Enter no of Process:" << endl;
    cin >> n;

    int arrival_time[n], burst_time[n], turn_around_time[n], waiting_time[n], process_seq[n];

    // Taking Input
    for (i = 0; i < n; i++)
    {
        cout << "\nEnter Arrival time " << i + 1 << " :";
        cin >> arrival_time[i];
        cout << "\nEnter Burst time " << i + 1 << " :";
        cin >> burst_time[i];
        process_seq[i] = i + 1;
    }

    // Displaying Input
    cout << "\n\n Before Scheduling:";
    cout << "\n ProcessId \t ArrivalTime \t BurstTime\n";
    for (i = 0; i < n; i++)
    {
        cout << process_seq[i] << "\t\t" << arrival_time[i] << "\t\t" << burst_time[i] << endl;
    }

    // Sort According to Burst Time
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n - 1; j++)
        {
            if (burst_time[j] > burst_time[j + 1])
            {
                temp = arrival_time[j];
                arrival_time[j] = arrival_time[j + 1];
                arrival_time[j + 1] = temp;

                temp = burst_time[j];
                burst_time[j] = burst_time[j + 1];
```

```
            burst_time[j + 1] = temp;

            temp = process_seq[j];
            process_seq[j] = process_seq[j + 1];
            process_seq[j + 1] = temp;
        }
    }
}

// Getting Minimum Arrival Time
int first_arrive = arrival_time[0], first_index_arrive = 0;

for (i = 0; i < n; i++)
{
    if (first_arrive > arrival_time[i])
    {
        first_arrive = arrival_time[i];
        first_index_arrive = i;
    }
}

// Task:
//   1) Take the first arrival job and execute it
//   2)  From the next job check if it is arrived or
//       not if arrived execute that process

// Waiting Time = TurnAroundTime  - BurstTime
// TurnAroundTime = ExitTime  - ArrivalTime

// Taking first arrival job
int process = n;
i = 0; // first_index_arrive;
total_time_cpu = first_arrive;

cout << "\n\n firs" << first_arrive << " " << first_index_arrive;

cout << "\n\n After Sorting:" << endl;
for (i = 0; i < n; i++)
{
    cout << process_seq[i] << "\t\t" << arrival_time[i] << "\t\t" << burst_time[i] << "\t\t" << endl;
}

// Completed Process Set
set<int> completed_process;

cout << "\n i:" << i;
cout << "\n ProcessID \t ArrivalTime \t BurstTime \t TurnAroundTime \t WaitingTime \n";
int flag = 1;

while (process != 0)
{
    // If Process Arrived Run it
    if (arrival_time[i] <= total_time_cpu && completed_process.find(i) == completed_process.end())
    {
        total_time_cpu += burst_time[i];
        turn_around_time[i] = total_time_cpu - arrival_time[i];
        waiting_time[i] = turn_around_time[i] - burst_time[i];

        total_waiting_time += waiting_time[i];
```

```
            total_turnaround_time += turn_around_time[i];

            // Once Process Complete decrement it
            process--;
            completed_process.insert(i);

            cout << process_seq[i] << "\t\t" << arrival_time[i] << "\t\t" << burst_time[i] << "\t\t" <<
turn_around_time[i] << "\t\t\t" << waiting_time[i] << endl;
            i = 0;
            continue;
        }

        if (i >= n - 1)
        {
            i = 0;
        }
        else
        {
            i++;
        }
    }

    cout << "\n\n Average Waiting Time: " << total_waiting_time / n;
    cout << "\n\n Average TurnAround Time: " << total_turnaround_time / n << endl;

    return 0;
}
```

Output:

```
Enter Arrival time 5 :
Enter Burst time 5 :

 Before Scheduling:
 ProcessId       ArrivalTime     BurstTime
1               0               8
2               1               5
3               3               3
4               4               1
5               6               4


 firs0 4

 After Sorting:
4               4               1
3               3               3
5               6               4
2               1               5
1               0               8

 i:5
 ProcessID       ArrivalTime     BurstTime       TurnAroundTime          WaitingTime
1               0               8               8                       0
4               4               1               5                       4
3               3               3               9                       6
5               6               4               10                      6
2               1               5               20                      15


 Average Waiting Time: 6.2

 Average TurnAround Time: 10.4
```

Program : Shortest Job First Preemptive

```
 #include<stdio.h>

struct Process
{
   int id,WT,AT,BT,TAT;
   // WT - Waiting Time
   // AT - Arrival Time
   // BT - Burst Time
   // TAT - TurnAroundTime
};

int main()
{
   freopen("input.txt","r",stdin);
   int n,temp[10];
   int count=0,t=0,short_P;
   float total_WT=0, total_TAT=0,Avg_WT,Avg_TAT;

   printf("Enter the number of the process\n");
   scanf("%d",&n);

   struct Process a[10];

   printf("Enter the arrival time and burst time of the process\n");
   printf("AT WT\n");

   for(int i=0;i<n;i++)
   {
      a[i].id = i+1;
      scanf("%d%d",&a[i].AT,&a[i].BT);

      // copying the burst time in
      // a temp array for the further use
      // in calculation of WT
      temp[i]=a[i].BT;
   }

   printf("\n\n Input Data:");
   printf("\n\nID \t AT \t BT \n\n");
   for(int i =0;i<n;i++)
   {
      printf("%d \t %d \t %d\n",a[i].id,a[i].AT,a[i].BT);
   }

   // we initialize the burst time
   // of a process with the maximum
   a[9].BT=10000;

   // loop will be execute until all the process
```

```
// complete so we use count!= number of
// the process

// t = time
for(t=0;count!=n;t++)
{
   // for finding min burst
   // it is useful
   short_P=9;

   for(int i=0;i<n;i++)
   {
      // If Process is shortest
      // and process arrived and It is remain to execute i.e Burst time greater than 0
      // So assign that as shortest process
      if(a[i].BT<a[short_P].BT && (a[i].AT<=t && a[i].BT>0))
      {
         short_P=i;
      }
   }
   // Reduce burst time by 1 and check again
   a[short_P].BT = a[short_P].BT-1;

   // if any process is completed
   if(a[short_P].BT==0)
   {
      // one process complete
      a[short_P].WT= t + 1 - a[short_P].AT - temp[short_P];
      a[short_P].TAT= t + 1 - a[short_P].AT;

      struct Process a1;

      a1 = a[count];
      a[count] = a[short_P];
      a[short_P] = a1;

      count++;

      // total calculation
      total_WT=total_WT+a[short_P].WT;
      total_TAT=total_TAT+a[short_P].TAT;
   }
}

Avg_WT=total_WT/n;
Avg_TAT=total_TAT/n;


// printing of the answer
printf("\n\n After Applying SJF Preemptive:");
printf("\n\nId \t AT \t BT \t WT \t TAT\n");
for(int i=0;i<n;i++)
```

```
    {
        printf("%d\t%d\t%d\t%d\t%d\n",a[i].id,a[i].AT,temp[i],a[i].WT,a[i].TAT);
    }

    printf("Avg waiting time of the process is %f\n",Avg_WT);
    printf("Avg turn around time of the process %f\n",Avg_TAT);

}
```

Output:

```
(base) ┌──(atharva㉿Atharva)-[~/College-Prac/OS_Practicals/Assignments/3-SJF-FCFS-Round-Robin]
└─$ ./sjfpre
Enter the number of the process
Enter the arrival time and burst time of the process
AT WT


 Input Data:

ID       AT      BT

1        0       8
2        1       5
3        3       3
4        4       1
5        6       4


 After Applying SJF Preemptive:

Id       AT      BT      WT      TAT
4       4       8       0       1
2       1       5       1       6
3       3       3       4       7
5       6       1       4       8
1       0       4       17      21
Avg waiting time of the process is 4.400000
Avg turn around time of the process 6.800000
```

Program : Round-Robin Algorithm

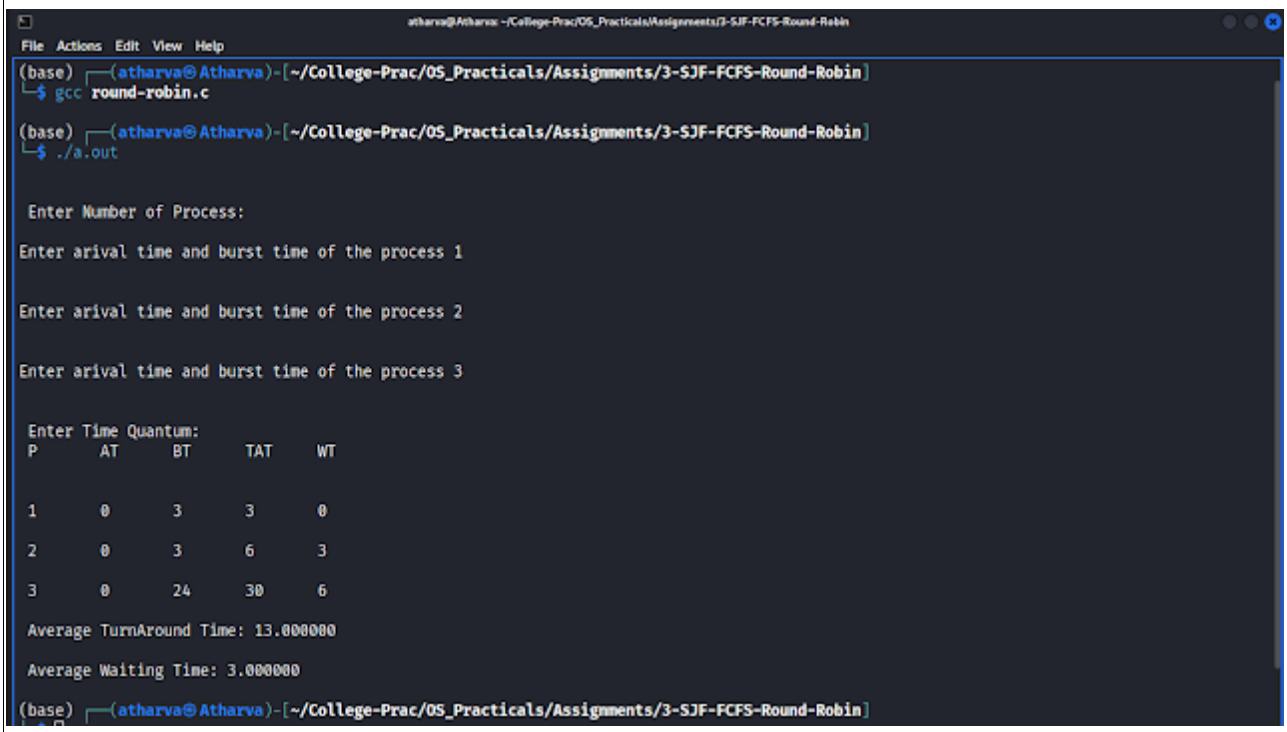
```
#include<stdio.h>

void main()
{
   freopen("input.txt","r",stdin);

   // Taking Input - NumberofProcess, Arival Time, Burst Time, Time Quantum

   int nop;
   printf("\n\n Enter Number of Process:");
   scanf("%d",&nop);

   int at[nop],bt[nop],temp[nop];

   for(int i=0;i<nop;i++)
   {
      printf("\n\nEnter arival time and burst time of the process %d \n",i+1);
      scanf("%d",&at[i]);
      scanf("%d",&bt[i]);

      // Storing Burst Time in temp array
      temp[i] = bt[i];
   }

   int time;
   printf("\n\n Enter Time Quantum:");
   scanf("%d",&time);

   // Using Round Robin Algoritm
   float avg_wt,avg_tat,wt=0,tat=0;
   int y = nop;
   int sum,count,i;

   printf("\n P \t AT \t BT \t TAT \t WT\n");


   sum = 0; // Storing burst time;
   i = 0; // For Pointing The Current Process

   while(y!=0)
   {
      // Checking if Process can complete in this Time Quantum
      if(temp[i] <= time && temp[i]>0)
      {
         sum = sum+temp[i]; // Storing Burst Time
         temp[i] = 0; // Process Complete
         count = 1;  // like flag to check process complete
      }
      // If Process take more time than 1 Time Quantum
```

```
      else if(temp[i] > 0)
      {
         temp[i] = temp[i] - time; // Decrementing the Quantum Time from Burst Time
         sum = sum + time;
      }

      // If Process Completes
      if(temp[i] == 0 && count == 1)
      {
         y--; // Decrementing Process Counter

         printf("\n\n %d \t %d \t %d\t %d\t %d",i+1,at[i],bt[i],sum-at[i],sum-at[i]-bt[i]);

         // TurnAroundTime = ExitTime - ArrivalTime
         // Waiting time = TurnAroundTime - BurstTime
         wt = wt + sum - at[i] - bt[i]; // Total waiting time
         tat = tat + sum -at[i]; // Total Burst Time
         count = 0;   // Resetting Flag
      }

      // printf("\n Current Index %d  Arrival Time %d  Remaining Burst Time %d",i,at[i],temp[i]);

      // If All Process Visited, then Run it in round robin fashion
      if(i == nop-1)
      {
         i=0;
      }
      else if(at[i+1] <= sum) // Checking if next process is arrived or not
      {
         i++;
      }
      // If process is not in end and the next process has not arrived yet so start the remaining
process again
      else
      {
         i = 0;
      }
   }


   // Calculating Average TurnAroundTime and Waiting Time
   avg_tat = tat/nop;
   avg_wt = wt/nop;

   printf("\n\n Average TurnAround Time: %f",avg_tat);
   printf("\n\n Average Waiting Time: %f\n",avg_wt);
}
/*
Input.txt File:
5

0 15
```

3 8
8 2
10 11
16 4

3
*/

Output:



Assignment 4A Producer-Consumer

Name: Atharva Agrawal
Roll No: 33303
Batch: L11

Code :

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <semaphore.h>
#include <unistd.h>

#define buffer_size 10

sem_t full, empty;
int buffer[buffer_size];
pthread_mutex_t mutex;
void *producer(void *p);
void *consumer(void *p);
void insert_item(int);
int remove_item();
int counter;
void initialize()
{
   pthread_mutex_init(&mutex, NULL);
   sem_init(&full, 1, 0);
   sem_init(&empty, 1, buffer_size);
   counter = 0;
}
int main()
{
   int n1, n2, i;
   printf("\nEnter no. of producers you want to create:");
   scanf("%d", &n1);
   printf("\nEnter no. of consumers you want to create:");
   scanf("%d", &n2);
   initialize();
   pthread_t tid[n1], tid1[n2];
   for (i = 0; i < n1; i++)
      pthread_create(&tid[i], NULL, producer, NULL);
   for (i = 0; i < n2; i++)
      pthread_create(&tid1[i], NULL, consumer, NULL);
   sleep(50);
   exit(0);
}
void *producer(void *p)
{
```

```
    int item, waittime;
    waittime = rand() % 5;
    sleep(waittime);
    item = rand() % 10;
    sem_wait(&empty);
    pthread_mutex_lock(&mutex);
    printf("\n Producer produced %d item", item);
    insert_item(item);
    pthread_mutex_unlock(&mutex);
    sem_post(&full);
}
void *consumer(void *p)
{
    int item, waittime;
    waittime = rand() % 10;
    sleep(
        waittime);
    sem_wait(&full);
    pthread_mutex_lock(&mutex);
    item = remove_item();
    printf("\n Consumer consumed %d item", item);
    pthread_mutex_unlock(&mutex);
    sem_post(&empty);
}
void insert_item(int item)
{
    buffer[counter++] = item;
}
int remove_item()
{
    return (buffer[--counter]);
}
```

Output:

```
(base) ┌──(atharva㉿Atharva)-[/media/…/Study/College-Prac/OS_Practicals/Assignments]
└─$ gcc 4-Producer-Consumer/4a.c   -lpthread

(base) ┌──(atharva㉿Atharva)-[/media/…/Study/College-Prac/OS_Practicals/Assignments]
└─$ ./a.out

Enter no. of producers you want to create:6

Enter no. of consumers you want to create:3

 Producer produced 3 item
 Producer produced 6 item
 Producer produced 7 item
 Consumer consumed 7 item
 Producer produced 0 item
 Producer produced 9 item
 Consumer consumed 9 item
 Producer produced 3 item
 Consumer consumed 3 item
(base) ┌──(atharva㉿Atharva)-[/media/…/Study/College-Prac/OS_Practicals/Assignments]
└─$
```

Assignment 4B Reader-Writer

Name: Atharva Agrawal
Roll No: 33303
Batch: L11

Code :

```
// Reader-Writer problem
#include <iostream>
#include <pthread.h>
#include <unistd.h>

using namespace std;

class monitor
{
private:
  int rcnt;   // no. of readers
  int wcnt; // no. of writers
  int waitr;   // no. of readers waiting
  int waitw;   // no. of writers waiting
  pthread_cond_t canread;   // condition variable to check whether reader can read
  pthread_cond_t canwrite;   // condition variable to check whether writer can write
  pthread_mutex_t condlock;   // mutex for synchronization

public:
  monitor()
  {
    rcnt = 0;
    wcnt = 0;
    waitr = 0;
    waitw = 0;
    pthread_cond_init(&canread, NULL);
    pthread_cond_init(&canwrite, NULL);
    pthread_mutex_init(&condlock, NULL);
  }
  // mutex provide synchronization so that no other thread
  // can change the value of data
  void beginread(int i)
  {
    pthread_mutex_lock(&condlock);
    // if there are active or waiting writers
    if (wcnt == 1 || waitw > 0)
    {
      // incrementing waiting readers
      waitr++;
      // reader suspended
      pthread_cond_wait(&canread, &condlock);
      waitr--;
    }
```

```
    // else reader reads the resource
    rcnt++;
    cout << "reader " << i << " is reading\n";
    pthread_mutex_unlock(&condlock);
    pthread_cond_broadcast(&canread);
  }
  void endread(int i)
  {
    // if there are no readers left then writer enters monitor
    pthread_mutex_lock(&condlock);
    if (--rcnt == 0)
      pthread_cond_signal(&canwrite);
    pthread_mutex_unlock(&condlock);
  }
  void beginwrite(int i)
  {
    pthread_mutex_lock(&condlock);
    // a writer can enter when there are no active
    // or waiting readers or other writer
    if (wcnt == 1 || rcnt > 0)
    {
      ++waitw;
      pthread_cond_wait(&canwrite, &condlock);
      --waitw;
    }
    wcnt = 1;
    cout << "writer " << i << " is writing\n";
    pthread_mutex_unlock(&condlock);
  }
  void endwrite(int i)
  {
    pthread_mutex_lock(&condlock);
    wcnt = 0;
    // if any readers are waiting, threads are unblocked
    if (waitr > 0)
      pthread_cond_signal(&canread);
    else
      pthread_cond_signal(&canwrite);
    pthread_mutex_unlock(&condlock);
  }
} M;

// global object of monitor class;
void *reader(void *id)
{
  int c = 0;
  int i = *(int *)id;
  // each reader attempts to read 5 times
  while (c < 5)
  {
    usleep(1);
    M.beginread(i);
```

```
    M.endread(i);
    c++;
  }
}
void *writer(void *id)
{
  int c = 0;
  int i = *(int *)id;
  // each writer attempts to write 5 times
  while (c < 5)
  {
    usleep(1);
    M.beginwrite(i);
    M.endwrite(i);
    c++;
  }
}

int main()
{
  pthread_t r[5], w[5];
  int id[5];
  for (int i = 0; i < 5; i++)
  {
    id[i] = i;
    // creating threads which execute reader function
    pthread_create(&r[i], NULL, &reader, &id[i]);
    // creating threads which execute writer function
    pthread_create(&w[i], NULL, &writer, &id[i]);
  }
  for (int i = 0; i < 5; i++)
  {
    pthread_join(r[i], NULL);
  }
  for (int i = 0; i < 5; i++)
  {
    pthread_join(w[i], NULL);
  }
}
```

Output:

```
(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/4-Producer-Consumer]
└─$ g++ -o s -pthread 4b.cpp

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/4-Producer-Consumer]
└─$ ./s
reader 0 is reading
writer 0 is writing
```

```
writer 1 is writing
reader 1 is reading
reader 2 is reading
writer 1 is writing
reader 2 is reading
writer 0 is writing
reader 2 is reading
reader 0 is reading
reader 1 is reading
writer 0 is writing
writer 1 is writing
writer 0 is writing
reader 0 is reading
writer 1 is writing
writer 2 is writing
reader 3 is reading
reader 2 is reading
writer 1 is writing
reader 1 is reading
writer 0 is writing
reader 2 is reading
reader 1 is reading
writer 3 is writing
reader 1 is reading
writer 2 is writing
reader 3 is reading
reader 0 is reading
writer 2 is writing
writer 4 is writing
writer 3 is writing
writer 2 is writing
reader 3 is reading
reader 0 is reading
writer 3 is writing
writer 2 is writing
writer 4 is writing
reader 4 is reading
reader 3 is reading
writer 3 is writing
writer 4 is writing
reader 4 is reading
reader 3 is reading
writer 4 is writing
writer 3 is writing
writer 4 is writing
reader 4 is reading
reader 4 is reading
reader 4 is reading

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/4-Producer-Consumer]
└─$
```

Assignment 5 Banker's Algorithm

Name: Atharva Agrawal
Roll No: 33303
Batch: L11

Code :

```
#include<stdio.h>

#include<stdlib.h>

/* Function to take input for 2D array */
void takeInput(int * a, int p, int r) {
  for (int i = 0; i < p; i++) {
    for (int j = 0; j < r; j++) {
      scanf("%d", & a[i * r + j]);
    }
  }
}

/* Safety function to check if the state of the system after request is safe or unsafe */
int safety(int * need, int * allocation, int work[], int finish[], int r, int p) {
  int safeseq[p];
  int count = 0;
  while (count < p) {
    int status = 0;
    for (int i = 0; i < p; i++) {
      if (finish[i] == 0) {
        int j = 0;
        for (j = 0; j < r; j++) {
          if (need[i * r + j] > work[j]) break;
        }
        if (j == r) {
          for (int k = 0; k < r; k++) {
            work[k] += allocation[i * r + k];
          }
          finish[i] = 1;
          safeseq[count++] = i;
          status = 1;
        }
      }
    }
    /* State is unsafe */
    if (status == 0) {
      return 0;
    }
  }
  /* State is safe and printing the safe sequence */
  printf("\nSAFE SEQUENCE");
  for (int i = 0; i < p; i++) {
```

```
    if (i == (p - 1)) printf(" %d ", safeseq[i]);
    else printf(" %d --> ", safeseq[i]);
  }
  return 1;
}
/* Bankers function */
void bankers(int * allocation, int available[], int * need, int r, int p) {
  int work[r];
  int finish[p];
  /* Initially setting finish for all process as false */
  for (int i = 0; i < p; i++) {
    finish[i] = 0;
  }
  printf("\nEnter the process number who requested resource :");
  int n = 3;
  scanf("%d", & n);
  int request[r];
  printf("Enter the request :");
  for (int i = 0; i < r; i++) {
    int a;
    scanf("%d", & a);
    request[i] = a;
  }
  // Check if the request <= need
  for (int i = 0; i < r; i++) {
    if (request[i] > need[n * r + i]) return;
  }
  // Check if the request <= available
  for (int i = 0; i < r; i++) {
    if (request[i] > available[i]) return;
  }
  /* Assuming that the request was granted */
  for (int i = 0; i < r; i++) {
    available[i] = available[i] - request[i];
    allocation[n * r + i] = allocation[n * r + i] + request[i];
    need[n * r + i] = need[n * r + i] - request[i];
  }
  for (int i = 0; i < r; i++) {
    work[i] = available[i];
  }
  /* Now checking if the state is safe or unsafe after request is granted */
  int res = safety((int * ) need, (int * ) allocation, work, finish, r, p);
  if (res) {
    printf("\nThe request of process %d for resource is safe\n", n);
  } else {
    printf("\nThe request of process %d for resource is not safe\n", n);
  }
  return;
}
/*Main function */
int main() {
  int r;
```

```
  printf("\nEnter total no of resources :");
  scanf("%d", & r);
  int p;
  printf("Enter total no of processes :");
  scanf("%d", & p);
  int allocation[p][r];
  int max[p][r];
  int need[p][r];
  int available[r];
  printf("\nEnter allocated resources for each process\n");
  takeInput((int * ) allocation, p, r);
  printf("\nEnter max resources needed for each process\n");
  takeInput((int * ) max, p, r);
  /* Calculating need matrix */
  for (int i = 0; i < p; i++) {
    for (int j = 0; j < r; j++) {
      int a = max[i][j];
      int b = allocation[i][j];
      need[i][j] = (a - b);
    }
  }
  printf("\nEnter available resources\n");
  for (int i = 0; i < r; i++) {
    int a;
    scanf("%d", & a);
    available[i] = a;
  }
  bankers((int * ) allocation, available, (int * ) need, r, p);
  return 0;
}
```

Output:

```
(base) ┌──(atharva㉿Atharva)-[/media/…/Study/College-Prac/OS_Practicals/Assignments]
└─$ cd 5-Banker-Algorithm

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/5-Banker-Algorithm]
└─$ gcc bankers.c

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/5-Banker-Algorithm]
└─$ ./a.out

Enter total no of resources :3
Enter total no of processes :5

Enter allocated resources for each process
1 2 3
3 0 1
2 2 0
1 3 1
0 2 3

Enter max resources needed for each process
4 4 8
7 1 4
4 3 2
7 5 4
2 5 5

Enter available resources
2 2 2

Enter the process number who requested resource :1
Enter the request :0 1 0

SAFE SEQUENCE 2 ⟶  4 ⟶  0 ⟶  1 ⟶  3
The request of process 1 for resource is safe

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/5-Banker-Algorithm]
└─$
```

Assignment 6 Page Replacement Algorithm

Name: Atharva Agrawal
Roll No: 33303
Batch: L11

FIFO (First In First Out)

Code :

```
#include <stdio.h>

int main() {
  int pageFaults = 0;
  int no_of_frames, m, n, s, no_of_pages;
  printf("Enter number of frames: ");
  scanf("%d", & no_of_frames);
  printf("Enter number of pages: ");
  scanf("%d", & no_of_pages);
  int incomingStream[no_of_pages];
  printf("Enter page reference string: ");
  for (int i = 0; i < no_of_pages; ++i) {
    scanf("%d", & incomingStream[i]);
  }
  int temp[no_of_frames];
  for (m = 0; m < no_of_frames; m++) {
    temp[m] = -1;
  }
  for (m = 0; m < no_of_pages; m++) {
    s = 0;
    for (n = 0; n < no_of_frames; n++) {
      if (incomingStream[m] == temp[n]) {
        s++;
        pageFaults--;
      }
    }
    pageFaults++;
    if ((pageFaults <= no_of_frames) && (s == 0)) {
      temp[m] = incomingStream[m];
    } else if (s == 0) {
      temp[(pageFaults - 1) % no_of_frames] = incomingStream[m];
    }
    printf("\n");
    printf("%d\t\t\t", incomingStream[m]);
    for (n = 0; n < no_of_frames; n++) {
      if (temp[n] != -1)
        printf(" %d\t", temp[n]);
      else
        printf(" - \t");
    }
```

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

Output:

```
(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/6-Page-Replacement Algorithms]
└─$ gcc fifo.c

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/6-Page-Replacement Algorithms]
└─$ ./a.out
Enter number of frames: 14
Enter number of pages: 14
Enter page reference string: 7 0 1 2 0 3 0 4 5 3 0 3 2 3

7                       7       -       -       -       -       -       -       -       -       -       -       -       -       -
0                       7       0       -       -       -       -       -       -       -       -       -       -       -       -
1                       7       0       1       -       -       -       -       -       -       -       -       -       -       -
2                       7       0       1       2       -       -       -       -       -       -       -       -       -       -
0                       7       0       1       2       -       -       -       -       -       -       -       -       -       -
3                       7       0       1       2       -       3       -       -       -       -       -       -       -       -
0                       7       0       1       2       -       3       -       -       -       -       -       -       -       -
4                       7       0       1       2       -       3       -       4       -       -       -       -       -       -
5                       7       0       1       2       -       3       -       4       5       -       -       -       -       -
3                       7       0       1       2       -       3       -       4       5       -       -       -       -       -
0                       7       0       1       2       -       3       -       4       5       -       -       -       -       -
3                       7       0       1       2       -       3       -       4       5       -       -       -       -       -
2                       7       0       1       2       -       3       -       4       5       -       -       -       -       -
3                       7       0       1       2       -       3       -       4       5       -       -       -       -       -
Total Page Faults:7

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/6-Page-Replacement Algorithms]
└─$
```

LRU (Least Recently Used)

Code:

```
#include<stdio.h>

int main() {
  int m, n, position, k, l, total_pages;
  int a = 0, b = 0, page_fault = 0;
  printf("Enter the no of pages\n");
  scanf("%d", & total_pages);
  int pages[total_pages];
  printf("Enter the page sequence:\n");
  for (int i = 0; i < total_pages; i++) {
    scanf("%d", & pages[i]);
  }
  int total_frames = 3;
  int frames[total_frames];
  int temp[total_frames];
  for (m = 0; m < total_frames; m++) {
    frames[m] = -1;
  }
  for (n = 0; n < total_pages; n++) {
    printf("%d: ", pages[n]);
    a = 0, b = 0;
    for (m = 0; m < total_frames; m++) {
      if (frames[m] == pages[n]) {
        a = 1;
        b = 1;
        break;
```

```
    }
   }
   if (a == 0) {
    for (m = 0; m < total_frames; m++) {
     if (frames[m] == -1) {
      frames[m] = pages[n];
      b = 1;
      page_fault++;
      break;
     }
    }
   }
   if (b == 0) {
    for (m = 0; m < total_frames; m++) {
     temp[m] = 0;
    }
    for (k = n - 1, l = 1; l <= total_frames - 1; l++, k--) {
     for (m = 0; m < total_frames; m++) {
      if (frames[m] == pages[k]) {
       temp[m] = 1;
      }
     }
    }
    for (m = 0; m < total_frames; m++) {
     if (temp[m] == 0)
      position = m;
    }
    frames[position] = pages[n];
    page_fault++;
   }
   for (m = 0; m < total_frames; m++) {
    if (frames[m] == -1) {
     printf("-\t");
    } else {
     printf("%d\t", frames[m]);
    }
   }
   printf("\n");
  }
  printf("\nTotal Number of Page Faults:\t%d\n", page_fault);
  return 0;
}
```

Output:

```
(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/6-Page-Replacement Algorithms]
└─$ gcc lru.c

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/6-Page-Replacement Algorithms]
└─$ ./a.out
Enter the no of pages
14
Enter the page sequence:
7 0 1 2 0 3 0 4 5 3 0 3 2 3
7: 7    -       -
0: 7    0       -
1: 7    0       1
2: 2    0       1
0: 2    0       1
3: 2    0       3
0: 2    0       3
4: 4    0       3
5: 4    0       5
3: 4    3       5
0: 0    3       5
3: 0    3       5
2: 0    3       2
3: 0    3       2

Total Number of Page Faults:    10

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/6-Page-Replacement Algorithms]
└─$
```

Optimal:

Code:

```
#include<stdio.h>

int main() {
  int no_of_frames, no_of_pages, frames[10], pages[30], temp[10], flag1,
    flag2, flag3, i, j, k, pos,
    max, faults = 0;
  printf("Enter number of frames: ");
  scanf("%d", & no_of_frames);
  printf("Enter number of pages: ");
  scanf("%d", & no_of_pages);
  printf("Enter page reference string: ");
  for (i = 0; i < no_of_pages; ++i) {
    scanf("%d", & pages[i]);
  }
  for (i = 0; i < no_of_frames; ++i) {
    frames[i] = -1;
  }
  for (i = 0; i < no_of_pages; ++i) {
    flag1 = flag2 = 0;
    for (j = 0; j < no_of_frames; ++j) {
      if (frames[j] == pages[i]) {
        flag1 = flag2 = 1;
        break;
      }
```

```
  }
  if (flag1 == 0) {
    for (j = 0; j < no_of_frames; ++j) {
      if (frames[j] == -1) {
        faults++;
        frames[j] = pages[i];
        flag2 = 1;
        break;
      }
    }
  }
  if (flag2 == 0) {
    flag3 = 0;
    for (j = 0; j < no_of_frames; ++j) {
      temp[j] = -1;
      for (k = i + 1; k < no_of_pages; ++k) {
        if (frames[j] == pages[k]) {
          temp[j] = k;
          break;
        }
      }
    }
    for (j = 0; j < no_of_frames; ++j) {
      if (temp[j] == -1) {
        pos = j;
        flag3 = 1;
        break;
      }
    }
    if (flag3 == 0) {
      max = temp[0];
      pos = 0;
      for (j = 1; j < no_of_frames; ++j) {
        if (temp[j] > max) {
          max = temp[j];
          pos = j;
        }
      }
    }
    frames[pos] = pages[i];
    faults++;
  }
  printf("\n");
  for (j = 0; j < no_of_frames; ++j) {
    if (frames[j] == -1) {
      printf("-\t");
    } else {
      printf("%d\t", frames[j]);
    }
  }
}
printf("\n\nTotal Page Faults = %d\n", faults);
```

```
 return 0;
}
```

Output:

```
(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/6-Page-Replacement Algorithms]
└─$ gcc optimal.c

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/6-Page-Replacement Algorithms]
└─$ ./a.out
Enter number of frames: 3
Enter number of pages: 14
Enter page reference string: 7 0 1 2 0 3 0 4 5 3 0 3 2 3

7       -       -
7       0       -
7       0       1
2       0       1
2       0       1
2       0       3
2       0       3
4       0       3
5       0       3
5       0       3
5       0       3
5       0       3
2       0       3
2       0       3

Total Page Faults = 8

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/6-Page-Replacement Algorithms]
└─$
```

Assignment 7 Inter Process Communication

Name: Atharva Agrawal
Roll No: 33303
Batch: L11

FIFO (First In First Out)

Code :
Program1.c:

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

int main() {
  int fd1, fd2; //fd is file descriptor
  char * f1loc = "myfifo1";
  char * f2loc = "myfifo2";
  char str[100], buffer[200];
  mkfifo(f1loc, 0666);
  printf("Enter a string: ");
  fgets(str, 100, stdin); //as string may contain spaces so to include that too
  fd1 = open(f1loc, O_WRONLY);
  if (write(fd1, str, strlen(str) + 1) == -1) //1 to include the /0 terminating character
  {
    printf("Error while writting to fifo 1 from process 1\n");
    return 1;
  }
  close(fd1);
  fd2 = open(f2loc, O_RDONLY);
  if (read(fd2, buffer, 150) == -1) {
    printf("Error while reading from fifo 2 from process 1\n");
    return 1;
  }
  close(fd2);
  printf("Message in FIFO 2 from Process 2:\n%s\n", buffer);
  return 0;
}
```

Program2.c:

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

int main()
{
  FILE * tfd;
  int fd1, fd2; //fd is file descriptor
  char * f1loc = "myfifo1";
  char * f2loc = "myfifo2";
  char buffer[100], tempstr[200];
  fd1 = open(f1loc, O_RDONLY);
  if (read(fd1, buffer, 100) == -1) {
    printf("Error while reading from fifo 1 from process 2\n");
    return 1;
  }
  close(fd1);
  printf("Message in FIFO 1 from Process 1:\n%s\n", buffer);
  int p = 0, l = 1, w = 1, c = 0;
  while (buffer[p] != '\0') {
    if (buffer[p] == '.') {
      l++;
    } else if (buffer[p] == ' ') {
      w++;
    }
    c++;
    p++;
  }
  c--; //to delete the read '\0'character
  printf("Information about the entered string\n");
  printf("No of lines: %d\n", l);
  printf("No of words: %d\n", w);
  printf("No of characters: %d\n", c);
  //preforming write on text file using high level system calls
  tfd = fopen("textfile.txt", "w");
  fprintf(tfd, "Data written in txt file\nInformation about the entered string\n");
  fprintf(tfd, "No of lines: %d\n", l);
  fprintf(tfd, "No of words: %d\n", w);
  fprintf(tfd, "No of characters: %d\n", c);
  fclose(tfd);
  //performing read from text file using high level system calls
  tfd = fopen("textfile.txt", "r");
```

```
 int i = 0;
 while (1) {
  if (feof(tfd)) //if chararacter is equal to end of file(EOF) break from the loop
  {
   break;
  }
  tempstr[i++] = fgetc(tfd);
 }
 tempstr[i++] = '\0';
 fclose(tfd);
 //creating and writting into second fifo file
 mkfifo(f2loc, 0666);
 fd2 = open(f2loc, O_WRONLY);
 if (write(fd2, tempstr, strlen(tempstr) + 1) == -1) //1 to include the /0 terminating character
 {
  printf("Error while writting to fifo 2 from process 2\n");
  return 1;
 }
 close(fd2);
 return 0;
}
```

Program1 Output:

```
(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/7-Inter process communication]
└─$ gcc program.c -o a.out

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/7-Inter process communication]
└─$ gcc program2.c -o b.out

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/7-Inter process communication]
└─$ ./a.out
Enter a string: Hello From Here
Message in FIFO 2 from Process 2:
Data written in txt file
Information about the entered string
No of lines: 1
No of words: 3
No of characters: 15
◆

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/7-Inter process communication]
└─$
```

Program2 Output:

```
(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/7-Inter process communication]
└─$ ./b.out
Message in FIFO 1 from Process 1:
Hello From Here

Information about the entered string
No of lines: 1
No of words: 3
No of characters: 15

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/7-Inter process communication]
└─$
```

Shared Memory

Code :

Server.c:

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

int main() {
  key_t key = ftok("shmfile", 65); //returns same key value in diffrent process when used
with
  //same file name and id
  int shmid;
  void * shmaddr;
  shmid = shmget(key, 100, 0666);
  printf("shmid of the shared memory is %d\n", shmid);
  shmaddr = shmat(shmid, NULL, 0);
  printf("the shared memory is attached to the address %p\n", shmaddr);
  printf("Data read from the shared memory is:\n%s", (char * ) shmaddr);
  return 0;
}
```

Client.c:

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

int main() {
  key_t key = ftok("shmfile", 65); //returns same key value in diffrent process when used
with
  //same file name and id
  int shmid;
  void * shmaddr;
  char str[100];
  shmid = shmget(key, 100, 0666 | IPC_CREAT);
  printf("shmid of the shared memory is %d\n", shmid);
  shmaddr = shmat(shmid, NULL, 0);
```

```
 printf("the shared memory is attached to the address %p\n", shmaddr);
 printf("Enter a string:\n");
 fgets(str, 100, stdin);
 strcpy(shmaddr, str);
 printf("Data written in shared memory is:\n%s", (char * ) shmaddr);
 return 0;
}
```

Server Output:

```
(base) ┌──(atharva㉿Atharva)-[/media/…/OS_Practicals/Assignments/7-Inter process communication/shared-memory]
└─$ ./b.out
shmid of the shared memory is 262184
the shared memory is attached to the address 0×7f142af16000
Data read from the shared memory is:

(base) ┌──(atharva㉿Atharva)-[/media/…/OS_Practicals/Assignments/7-Inter process communication/shared-memory]
└─$ ./b.out
shmid of the shared memory is 262184
the shared memory is attached to the address 0×7f0ea5a3e000
Data read from the shared memory is:
Hello, Atharva Agrawal here

(base) ┌──(atharva㉿Atharva)-[/media/…/OS_Practicals/Assignments/7-Inter process communication/shared-memory]
└─$
```

Client Output:

```
(base) ┌──(atharva㉿Atharva)-[/media/…/OS_Practicals/Assignments/7-Inter process communication/shared-memory]
└─$ gcc client.c -o a.out

(base) ┌──(atharva㉿Atharva)-[/media/…/OS_Practicals/Assignments/7-Inter process communication/shared-memory]
└─$ ./a.out
shmid of the shared memory is 262184
the shared memory is attached to the address 0×7fab2e797000
Enter a string:
Hello, Atharva Agrawal here
Data written in shared memory is:
Hello, Atharva Agrawal here

(base) ┌──(atharva㉿Atharva)-[/media/…/OS_Practicals/Assignments/7-Inter process communication/shared-memory]
└─$
```

Assignment 8 Disk-Scheduling-Algorithm

Name: Atharva Agrawal
Roll No: 33303
Batch: L11

SSTF (First In First Out)

Code :

```
#include<stdio.h>

#include<stdlib.h>

void sort(int * arr, int n) {
  int temp;
  for (int i = 0; i < n - 1; i++) {
    for (int j = i + 1; j < n; j++) {
      if (arr[i] > arr[j]) {
        temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
      }
    }
  }
}

int main() {
  int n, headpos;
  printf("Enter the no of disk scheduling requests: ");
  scanf("%d", & n);
  printf("Enter the initial head position: ");
  scanf("%d", & headpos);
  int arr[n + 1], templ[n], tempr[n];
  int p = 0, q = 0;
  printf("Enter the elements in the queue\n");
  for (int i = 0; i < n; i++) {
    scanf("%d", & arr[i]);
    if (arr[i] <= headpos) {
      templ[p++] = arr[i];
    } else {
      tempr[q++] = arr[i];
    }
  }
  sort(templ, p);
  sort(tempr, q);
  int temparr[50];
  int k = 1;
  temparr[0] = headpos;
  int lind = p - 1, rind = 0, templele, temprele, curr_head = headpos;
```

```
 while (lind >= 0 && rind != q - 1) {
   templele = templ[lind];
   temprele = tempr[rind];
   if (abs(templele - curr_head) < abs(temprele - curr_head)) {
     temparr[k++] = templele;
     lind--;
     curr_head = templele;
   } else {
     temparr[k++] = temprele;
     rind++;
     curr_head = temprele;
   }
 }
 while (lind > 0) {
   temparr[k++] = templ[lind--];
 }
 while (rind < q) {
   temparr[k++] = tempr[rind++];
 }
 float totalseek = 0;
 int diff;
 for (int i = 0; i < k - 1; i++) {
   diff = abs(temparr[i] - temparr[i + 1]);
   totalseek += diff;
   printf("Disk head moves form %d to %d with a seek of %d\n", temparr[i], temparr[i + 1], diff);
 }

 float avgseek = (totalseek / n);
 printf("Total seek time: %.2f \n", totalseek);
 printf("Average seek time: %.2f \n", avgseek);
 return 0;
}
```

Output:

```
(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/8-Inter-process Communication using Shared Memory using System
└─$ gcc sstf.c

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/8-Inter-process Communication using Shared Memory using System
└─$ ./a.out
Enter the no of disk scheduling requests: 7
Enter the initial head position: 50
Enter the elements in the queue
82
170
43
140
24
16
190
Disk head moves form 50 to 43 with a seek of 7
Disk head moves form 43 to 24 with a seek of 19
Disk head moves form 24 to 16 with a seek of 8
Disk head moves form 16 to 82 with a seek of 66
Disk head moves form 82 to 140 with a seek of 58
Disk head moves form 140 to 170 with a seek of 30
Disk head moves form 170 to 190 with a seek of 20
Total seek time: 208.00
Average seek time: 29.71

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/8-Inter-process Communication using Shared Memory using System
└─$
```

SCAN

Code :

```
#include<stdio.h>

#include<stdlib.h>

void sort(int * arr, int n) {
  int temp;
  for (int i = 0; i < n - 1; i++) {
    for (int j = i + 1; j < n; j++) {
      if (arr[i] > arr[j]) {
        temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
      }
    }
  }
}
int main() {
  int n, maxrange, headpos, drn;
  printf("Enter the no of disk scheduling requests: ");
  scanf("%d", & n);
  printf("Enter the maximum range of the disk: ");
  scanf("%d", & maxrange);
  printf("Enter the initial head position: ");
  scanf("%d", & headpos);
  printf("Enter the intial direction of the disk \n1)Left\n2)right\nEnter your choice: ");
  scanf("%d", & drn);
  int arr[n + 1], templ[n], tempr[n];
  int p = 0, q = 0;
  printf("Enter the elements in the queue\n");
  for (int i = 0; i < n; i++) {
    scanf("%d", & arr[i]);
    if (arr[i] <= headpos) {
      templ[p++] = arr[i];
    } else {
      tempr[q++] = arr[i];
    }
  }
  sort(templ, p);
  sort(tempr, q);
  int temparr[50];
  int k = 1;
  if (drn == 1) {
    temparr[0] = headpos;
    for (int i = p - 1; i >= 0; i--) {
      temparr[k++] = templ[i];
    }
    temparr[k++] = 0;
```

```
    for (int i = 0; i < q; i++) {
      temparr[k++] = tempr[i];
    }
  } else {
    temparr[0] = headpos;
    for (int i = 0; i < q; i++) {
      temparr[k++] = tempr[i];
    }
    temparr[k++] = maxrange;
    for (int i = p - 1; i >= 0; i--) {
      temparr[k++] = templ[i];
    }
  }
  float totalseek = 0;
  int diff;
  for (int i = 0; i < k - 1; i++) {
    diff = abs(temparr[i] - temparr[i + 1]);
    totalseek += diff;
    printf("Disk head moves form %d to %d with a seek of %d\n", temparr[i], temparr[i + 1], diff);
  }
  printf("The sequence is:\n");
  for (int i = 0; i < k - 1; i++) {
    printf("%d -> ", temparr[i]);
  }
  printf("%d\n", temparr[k - 1]);
  float avgseek = (totalseek / n);
  printf("Total seek time: %.2f \n", totalseek);
  printf("Average seek time: %.2f \n", avgseek);
  return 0;
}
```

Output:

```
(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/8-Inter-process Communication using Shared Memory using System]
└─$ gcc scan.c

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/8-Inter-process Communication using Shared Memory using System]
└─$ ./a.out
Enter the no of disk scheduling requests: 5
Enter the maximum range of the disk: 100
Enter the initial head position: 40
Enter the intial direction of the disk
1)Left
2)right
Enter your choice: 1
Enter the elements in the queue
82 150 43 130 20
Disk head moves form 40 to 20 with a seek of 20
Disk head moves form 20 to 0 with a seek of 20
Disk head moves form 0 to 43 with a seek of 43
Disk head moves form 43 to 82 with a seek of 39
Disk head moves form 82 to 130 with a seek of 48
Disk head moves form 130 to 150 with a seek of 20
The sequence is:
40 → 20 → 0 → 43 → 82 → 130 → 150
Total seek time: 190.00
Average seek time: 38.00

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/8-Inter-process Communication using Shared Memory using System]
└─$
```

CLOOK

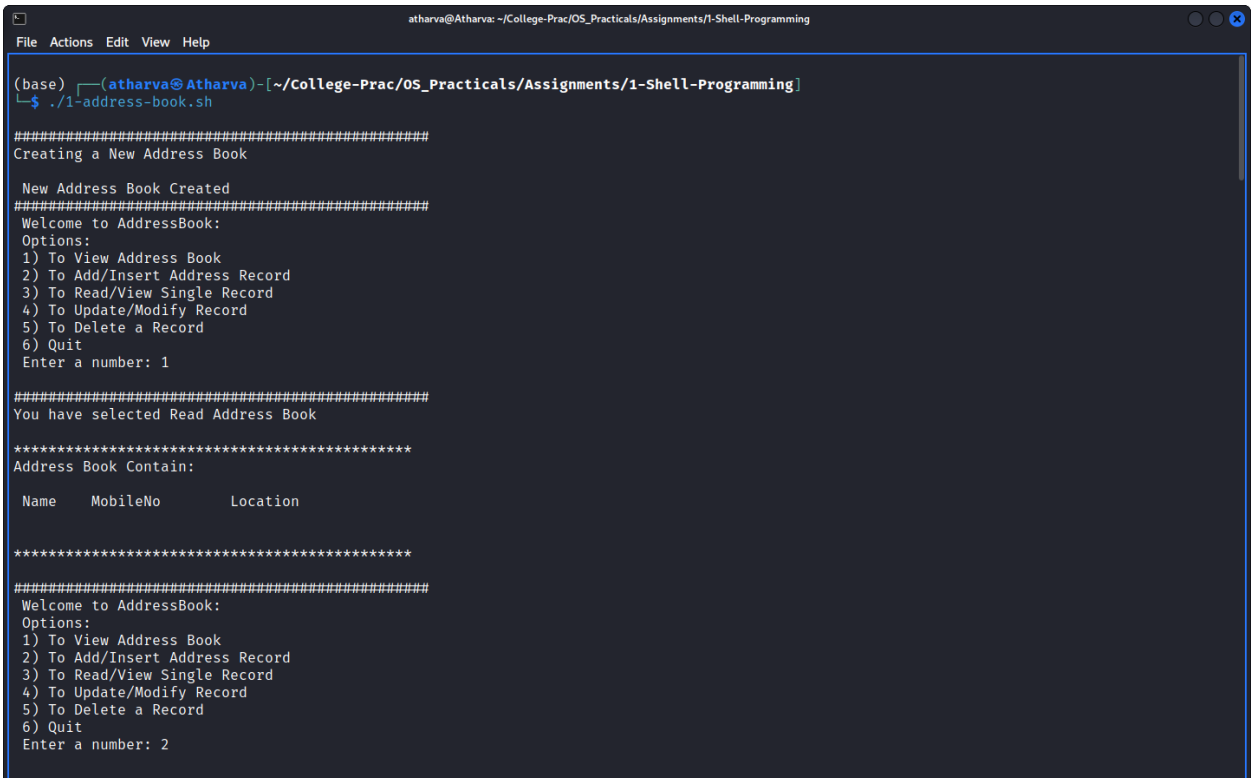
```
#include<stdio.h>
#include<stdlib.h>

void sort(int * arr, int n) {
  int temp;
  for (int i = 0; i < n - 1; i++) {
    for (int j = i + 1; j < n; j++) {
      if (arr[i] > arr[j]) {
        temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
      }
    }
  }
}

int main()
{
    int n, maxrange, headpos, drn;

    printf("Enter the no of disk scheduling requests: ");
    scanf("%d", & n);
    printf("Enter the maximum range of the disk: ");
    scanf("%d", & maxrange);
    printf("Enter the initial head position: ");
    scanf("%d", & headpos);
    printf("Enter the intial direction of the disk \n1)Left\n2)right\nEnter your choice: ");
    scanf("%d", & drn);

    int arr[n + 1], templ[n], tempr[n];
    int p = 0, q = 0;

    printf("Enter the elements in the queue\n");

    for (int i = 0; i < n; i++)
    {
        scanf("%d", & arr[i]);
        if (arr[i] <= headpos)
        {
          templ[p++] = arr[i];
        } else
        {
          tempr[q++] = arr[i];
        }
    }

    sort(templ, p);
    sort(tempr, q);

    int temparr[50];
    int k = 1;
```

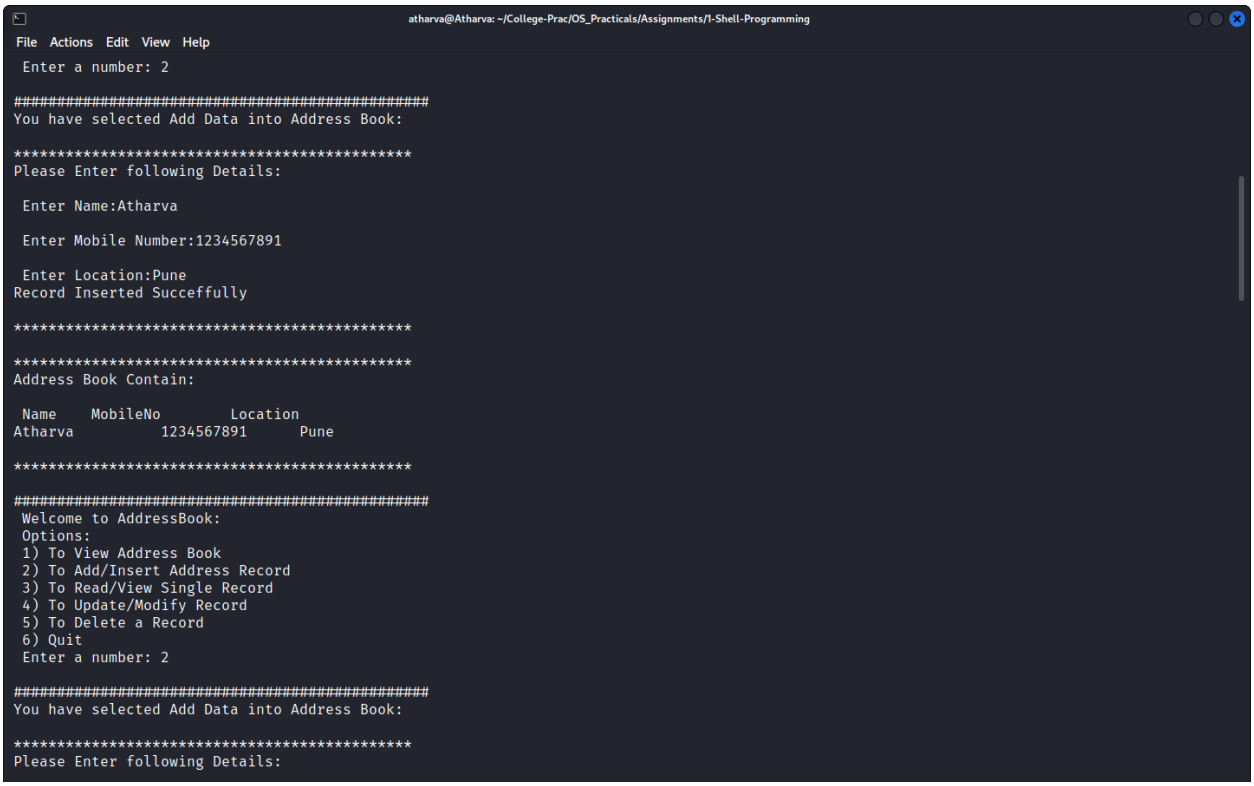
```
    if (drn == 1)
    {
        temparr[0] = headpos;

        for (int i = p - 1; i >= 0; i--) {
            temparr[k++] = templ[i];
        }
        for (int i = q - 1; i >= 0; i--) {
            temparr[k++] = tempr[i];
        }
    }
    else
    {
        temparr[0] = headpos;

        for (int i = 0; i < q; i++) {
            temparr[k++] = tempr[i];
        }

        for (int i = 0; i < p; i++) {
            temparr[k++] = templ[i];
        }
    }

    float totalseek = 0; int diff;
    for (int i = 0; i < k - 1; i++)
    {
        diff = abs(temparr[i] - temparr[i + 1]);
        totalseek += diff;
        printf("Disk head moves form %d to %d with a seek of %d \n ",temparr[i],temparr[i+1],diff);
    }

    float avgseek = (totalseek / n);
    printf("Total seek time: %.2f \n", totalseek);
    printf("Average seek time: %.2f \n", avgseek);

    return 0;
}
```

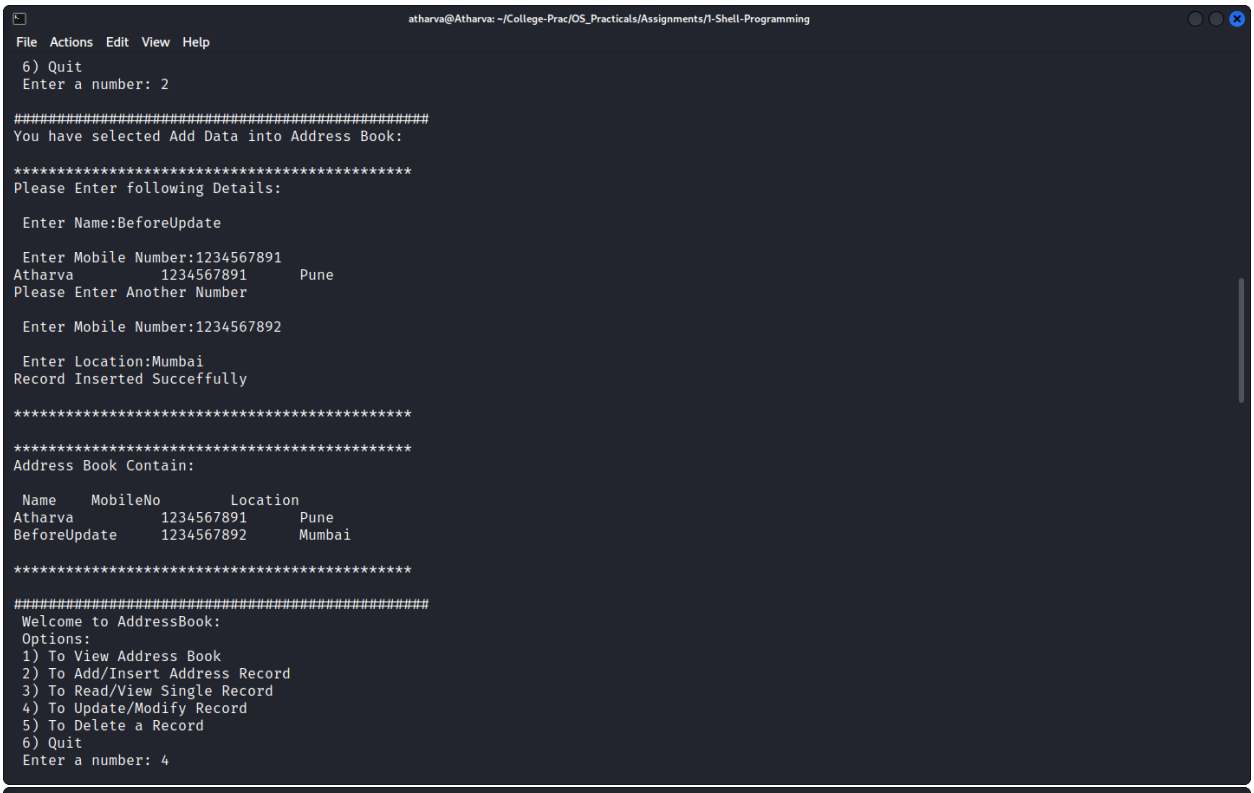
Output:

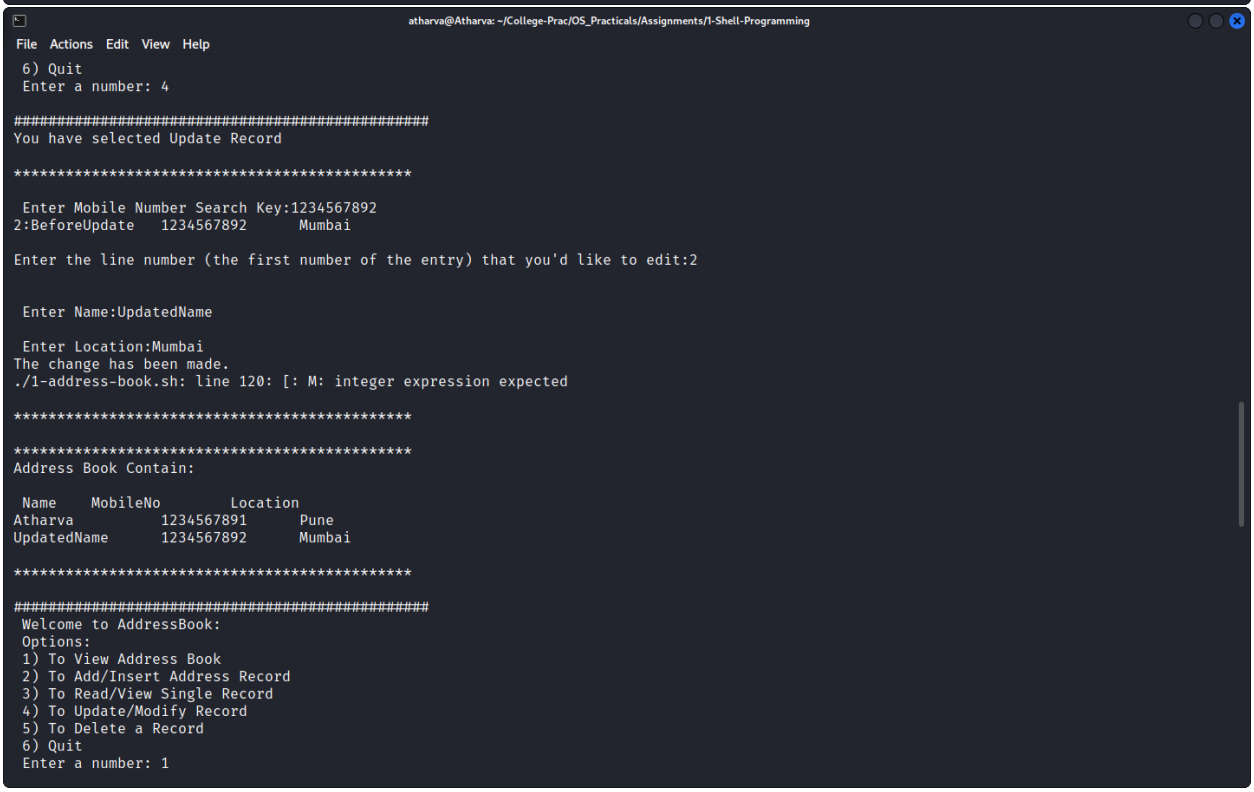
```
(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/8-Inter-process Communication using Shared Memory using System]
└─$ gcc clook.c

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/8-Inter-process Communication using Shared Memory using System]
└─$ ./a.out
Enter the no of disk scheduling requests: 7
Enter the maximum range of the disk: 199
Enter the initial head position: 60
Enter the intial direction of the disk
1)Left
2)right
Enter your choice: 2
Enter the elements in the queue
95
20
100
130
19
59
35
Disk head moves form 60 to 95 with a seek of 35
 Disk head moves form 95 to 100 with a seek of 5
 Disk head moves form 100 to 130 with a seek of 30
 Disk head moves form 130 to 19 with a seek of 111
 Disk head moves form 19 to 20 with a seek of 1
 Disk head moves form 20 to 35 with a seek of 15
 Disk head moves form 35 to 59 with a seek of 24
 Total seek time: 221.00
Average seek time: 31.57

(base) ┌──(atharva㉿Atharva)-[/media/…/College-Prac/OS_Practicals/Assignments/8-Inter-process Communication using Shared Memory using System]
└─$ clear
```

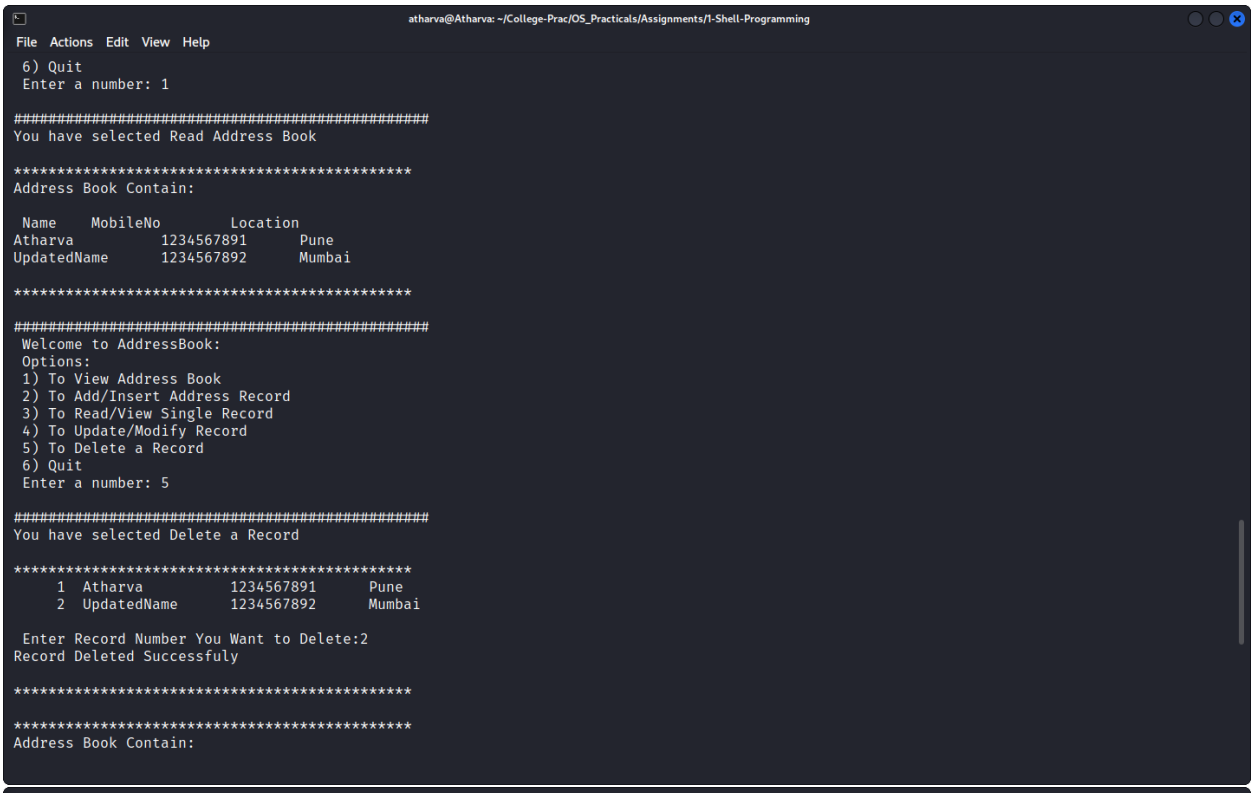
Output 1B: AddressBook using Shell Script

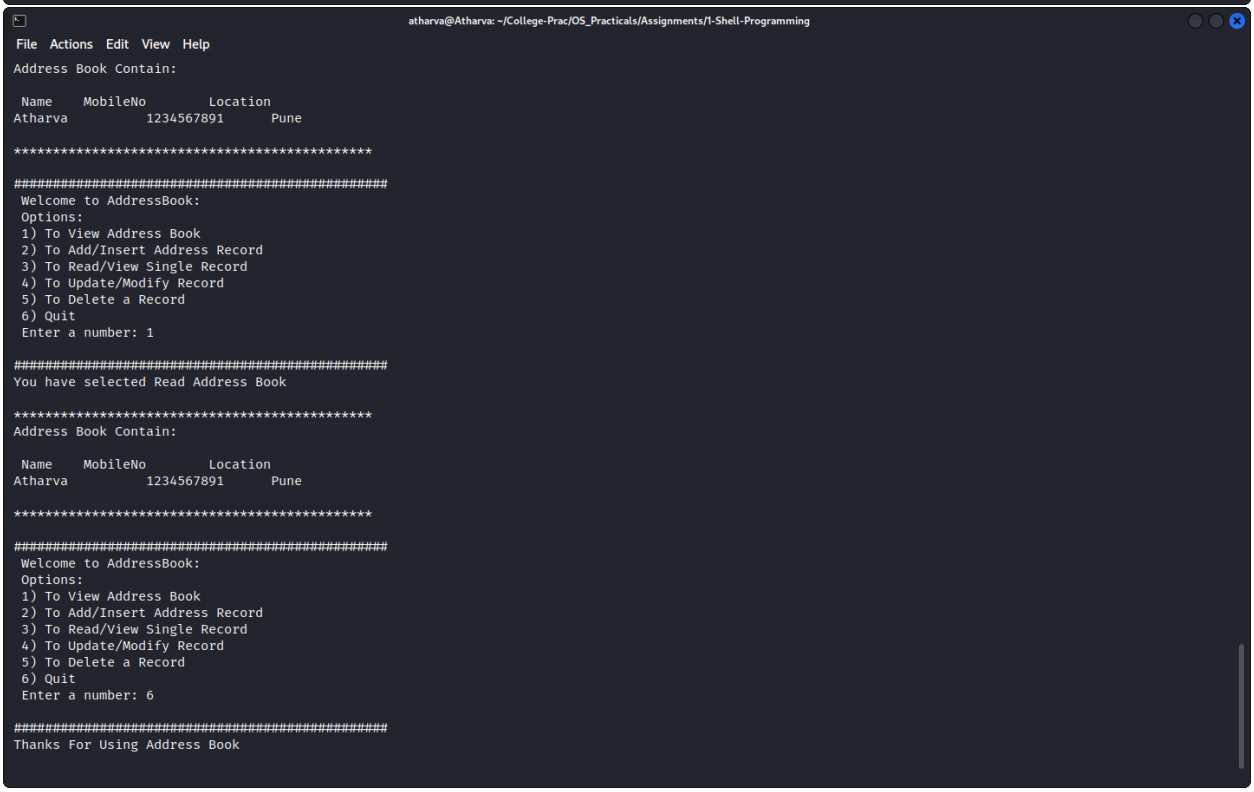





atharva@Atharva: ~/College-Prac/OS_Practicals/Assignments/1-Shell-Programming
File Actions Edit View Help
6) Quit
Enter a number: 2
##
You have selected Add Data into Address Book:
****
Please Enter following Details:
Enter Name:BeforeUpdate
Enter Mobile Number:1234567891
Atharva 1234567891 Pune
Please Enter Another Number
Enter Mobile Number:1234567892
Enter Location:Mumbai
Record Inserted Succeffully
****
****
Address Book Contain:
Name MobileNo Location
Atharva 1234567891 Pune
BeforeUpdate 1234567892 Mumbai
****
##
Welcome to AddressBook:
Options:
1) To View Address Book
2) To Add/Insert Address Record
3) To Read/View Single Record
4) To Update/Modify Record
5) To Delete a Record
6) Quit
Enter a number: 4


atharva@Atharva: ~/College-Prac/OS_Practicals/Assignments/1-Shell-Programming
File Actions Edit View Help
6) Quit
Enter a number: 4
##
You have selected Update Record
****
Enter Mobile Number Search Key:1234567892
2:BeforeUpdate 1234567892 Mumbai
Enter the line number (the first number of the entry) that you'd like to edit:2
Enter Name:UpdatedName
Enter Location:Mumbai
The change has been made.
./1-address-book.sh: line 120: [: M: integer expression expected
****
****
Address Book Contain:
Name MobileNo Location
Atharva 1234567891 Pune
UpdatedName 1234567892 Mumbai
****
##
Welcome to AddressBook:
Options:
1) To View Address Book
2) To Add/Insert Address Record
3) To Read/View Single Record
4) To Update/Modify Record
5) To Delete a Record
6) Quit
Enter a number: 1


atharva@Atharva: ~/College-Prac/OS_Practicals/Assignments/1-Shell-Programming
File Actions Edit View Help
6) Quit
Enter a number: 1
##
You have selected Read Address Book
****
Address Book Contain:
Name MobileNo Location
Atharva 1234567891 Pune
UpdatedName 1234567892 Mumbai
****
##
Welcome to AddressBook:
Options:
1) To View Address Book
2) To Add/Insert Address Record
3) To Read/View Single Record
4) To Update/Modify Record
5) To Delete a Record
6) Quit
Enter a number: 5
##
You have selected Delete a Record
****
1 Atharva 1234567891 Pune
2 UpdatedName 1234567892 Mumbai
Enter Record Number You Want to Delete:2
Record Deleted Successfuly
****
****
Address Book Contain:


atharva@Atharva: ~/College-Prac/OS_Practicals/Assignments/1-Shell-Programming
File Actions Edit View Help
Address Book Contain:
Name MobileNo Location
Atharva 1234567891 Pune
****
##
Welcome to AddressBook:
Options:
1) To View Address Book
2) To Add/Insert Address Record
3) To Read/View Single Record
4) To Update/Modify Record
5) To Delete a Record
6) Quit
Enter a number: 1
##
You have selected Read Address Book
****
Address Book Contain:
Name MobileNo Location
Atharva 1234567891 Pune
****
##
Welcome to AddressBook:
Options:
1) To View Address Book
2) To Add/Insert Address Record
3) To Read/View Single Record
4) To Update/Modify Record
5) To Delete a Record
6) Quit
Enter a number: 6
##
Thanks For Using Address Book