Operating Systems

Lab Manual

Submitted in the partial fulfillment of the
requirements for the award of Degree of

Bachelor of Engineering

in

Computer Science and Engineering

By

NAME: Sriharini Margapuri

ROLL NO: 1005-21-733065

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY COLLEGE OF ENGINEERING (A)
Osmania University, Hyderabad – 500 007 2022-2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY COLLEGE OF ENGINEERING (A)
Osmania University, Hyderabad – 500 007

CERTIFICATE

This is to certify that **Sriharini Margapuri** bearing

Roll no: **1005-21-733065** studying B.E. V Semester has successfully completed

"**Operating Systems Lab**" for the academic year 2023-24.

Internal Examiner **External Examiner**

TABLE OF CONTENTS

TABLE OF CONTENTS

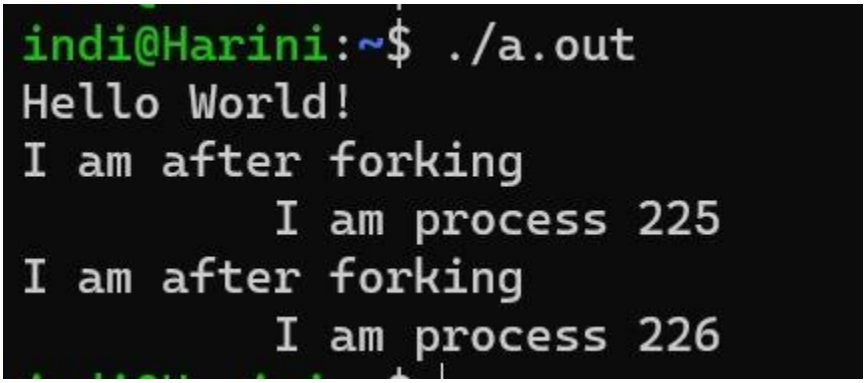
Program 1: Write a program to demonstrate open(), close(), read(), write() system calls.

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

static char message[]="Hello world";

int main(){
    int fd;
    char *buffer[80];
    fd=open("datafile.dat",O_RDWR|O_CREAT|O_EXCL,S_IREAD|S_IWRITE);
    if(fd!=-1){
        printf("File is opened for read/write access\n");
        write(fd,message,sizeof(message));
        close(fd);
        fd=open("datafile.dat",O_RDWR,S_IREAD|S_IWRITE);
        if(read(fd,buffer,sizeof(message))==sizeof(message)){
            printf("\"  %s\"  was written to datafile.dat\n",buffer);
        }else{
            printf("*** error reading datafile.dat ***");
        }
        close(fd);
    }else{
        printf("datafile.dat already exists\n");
        fd=open("datafile.dat",O_RDWR,S_IREAD|S_IWRITE);
        if(read(fd,buffer,sizeof(message))==sizeof(message)){
            printf("\"  %s\"  was written to datafile.dat\n",buffer);
        }else{
            printf("*** Error reading datafile.dat ***");
        }
        close(fd);
    }
    exit(0);
}
```

```
PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab> cd "c:\Use
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc systemcall.c -o systemcall }
File is opened for read/write access
"  Hello world"  was written to datafile.dat
```

Program 2: Write a program to demonstrate fork() system call.

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

int main(void){
    printf("Hello World!\n");
    fork();
    printf("I am after forking\n");
    printf("\t I am process %d \n", getpid());
}
```



Program 3: Write a program to demonstrate multiple fork().

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

int main(void){
    printf("Here I'm just before 1st forking statement\n");
    fork();
    printf("Here I'm just after 1st forking statement\n");
    fork();
    printf("Here I'm just after 2nd forking statement\n");
    fork();
    printf("Here I'm just after 3rd forking statement\n");
    printf("\tHello world from process %d\n", getpid());
}
```

```
indi@Harini:~$ cc multiplefork.c
indi@Harini:~$ ./a.out
Here I'm just before 1st forking statement
Here I'm just after 1st forking statement
Here I'm just after 1st forking statement
Here I'm just after 2nd forking statement
Here I'm just after 2nd forking statement
Here I'm just after 2nd forking statement
Here I'm just after 2nd forking statement
Here I'm just after 3rd forking statement
Here I'm just after 3rd forking statement
        Hello world from process 234
        Hello world from process 233
Here I'm just after 3rd forking statement
Here I'm just after 3rd forking statement
Here I'm just after 3rd forking statement
        Hello world from process 237
        Hello world from process 235
        Hello world from process 238
Here I'm just after 3rd forking statement
Here I'm just after 3rd forking statement
        Hello world from process 239
        Hello world from process 236
Here I'm just after 3rd forking statement
        Hello world from process 240
```

Program 4: Write a program to demonstrate pipe:

A. Parent Writing, Child Reading

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

#define SIZE 1024

int main(){
    int pfd[2],nread,pid;
    char buf[SIZE];
    if(pipe(pfd)==-1){
        perror("Pipe failed");
        exit(1);
    }
    if((pid=fork())<0){
        perror("Fork failed");
        exit(2);
    }
    if(pid==0){
        close(pfd[1]);
        while((nread=read(pfd[0],buf,SIZE))!=0)
        printf("Child read %s\n",buf);
        close(pfd[0]);
    }else{
        close(pfd[0]);
        strcpy(buf,"Hello...");
        write(pfd[1],buf,strlen(buf)+1);
        close(pfd[1]);
    }
}
```

```
indi@Harini:~$ ./a.out
Child read Hello...
```

B. Child Writing, Parent Reading

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

#define READ 0
#define WRITE 1

char *phrase="Welcome to OS lab";

int main(){
    int fd[2],bytesread;
    char message[100];
    pipe(fd);
    if(fork()==0){
```

```
15          close(fd[READ]);
16          write(fd[WRITE],phrase,strlen(phrase)+1);
17          close(fd[WRITE]);
18      }else{
19          close(fd[WRITE]);
20          bytesread=read(fd[READ],message,100);
21          printf("Read %d bytes: %s\n",bytesread,message);
22          close(fd[READ]);
23      }
24  }
```

```
indi@Harini:~$ ./a.out
Read 18 bytes: Welcome to OS lab
indi@Harini:~$
```

Program 5: Write a program for FCFS (non-preemptive) scheduling algorithm.

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

int main(){
    int i, n;
    float sumw = 0, sumt = 0;
    int w[10], b[10], t[10], p[10];
    float avgw, avgt;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    for(i=0; i<n; i++){
        printf("Enter the burst time for process %d: ", (i+1));
        scanf("%d", &b[i]);
        p[i] = i+1;
    }
    w[0]=0;
    printf("\tProcess \t Burst Time \t Waiting Time \t Turnaround Time\n");
    for(i=0; i<n; i++){
        t[i] = b[i] + w[i];
        printf("\t\t %d \t\t %d \t\t %d \t\t %d\n", p[i], b[i], w[i],
t[i]);
        w[i+1] = w[i] + b[i];
        sumw += w[i];
        sumt += t[i];
    }
    avgw = sumw/n;
    avgt = sumt/n;
    printf("Average waiting time: %f\n", avgw);
    printf("Average turnaround time: %f\n", avgt);
}
```

```
PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc fcfs.c -o fcfs } ;
Enter the number of processes: 3
Enter the burst time for process 1: 24
Enter the burst time for process 2: 3
Enter the burst time for process 3: 3
        Process         Burst Time      Waiting Time    Turnaround Time
                1               24              0               24
                2               3               24              27
                3               3               27              30
Average waiting time: 17.000000
Average turnaround time: 27.000000
```

Program 6: Write a program for FCFS (preemptive) scheduling algorithm.

```
#include<stdio.h>
#include<stdlib.h>
int main(){
    int i, j, n, temp;
    float sumw = 0, sumt = 0;
    int w[10], b[10], t[10], p[10], a[10];
    float avgw, avgt;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
     for(i=0; i<n; i++){
        printf("Enter the burst time for process %d: ", (i+1));
        scanf("%d", &b[i]);
        printf("Enter the arrival time for process %d: ", (i+1));
        scanf("%d", &a[i]);
        p[i] = i+1;
    }
    for(i=0; i<n; i++){
        for(j=i+1; j<n; j++){
            if(a[i] > a[j]){
                temp = b[i];
                b[i] = b[j];
                b[j] = temp;

                temp = p[i];
                p[i] = p[j];
                p[j] = temp;

                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }
    w[0]=0;
    printf("\tProcess \t Burst Time \t Arrival Time \t Waiting Time \t
Turnaround Time\n");
    for(i=0; i<n; i++){
        t[i] = b[i] + w[i]-a[i];
        printf("\t\t %d \t\t %d \t\t %d \t\t %d \t\t %d\n", p[i], b[i],
a[i], w[i], t[i]);
        w[i+1] = w[i] + b[i] - a[i];
        sumw += w[i];
        sumt += t[i];
    }
    avgw = sumw/n;
    avgt = sumt/n;
    printf("Average waiting time: %f\n", avgw);
    printf("Average turnaround time: %f\n", avgt);
}
```

```
Average turnaround time: 89500284.000000
PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab> cd "c:\Users\s
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc fcfsp.c -o fcfsp } ; if ($?) { .\
Enter the number of processes: 3
Enter the burst time for process 1: 24
Enter the arrival time for process 1: 2
Enter the burst time for process 2: 3
Enter the arrival time for process 2: 0
Enter the burst time for process 3: 3
Enter the arrival time for process 3: 1
        Process          Burst Time      Arrival Time    Waiting Time    Turnaround Time
                2               3               0               0               3
                3               3               1               3               5
                1               24              2               5               27
Average waiting time: 2.666667
Average turnaround time: 11.666667
```

Program 7: Write a program for SJF (non-preemptive) algorithm.

```
#include<stdio.h>
#include<stdlib.h>
int main(){
    int i, j, n, temp;
    float sumw = 0, sumt = 0;
    int w[10], b[10], t[10], p[10];
    float avgw, avgt;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
     for(i=0; i<n; i++){
        printf("Enter the burst time for process %d: ", (i+1));
        scanf("%d", &b[i]);
        p[i] = i+1;
    }
    for(i=0; i<n; i++){
        for(j=i+1; j<n; j++){
            if(b[i] > b[j]){
                temp = b[i];
                b[i] = b[j];
                b[j] = temp;

                temp = p[i];
                p[i] = p[j];
                p[j] = temp;
            }
        }
    }
    w[0]=0;
    printf("\tProcess \t Burst Time \t Waiting Time \t Turnaround Time\n");
    for(i=0; i<n; i++){
        t[i] = b[i] + w[i];
        printf("\t\t %d \t\t %d \t\t %d \t\t %d\n", p[i], b[i], w[i],
t[i]);
        w[i+1] = w[i] + b[i];
        sumw += w[i];
        sumt += t[i];
    }
    avgw = sumw/n;
    avgt = sumt/n;
    printf("Average waiting time: %f\n", avgw);
    printf("Average turnaround time: %f\n", avgt);
}
```

```
PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab>
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc sjfnp.c -o sjfnp }
Enter the number of processes: 3
Enter the burst time for process 1: 24
Enter the burst time for process 2: 3
Enter the burst time for process 3: 3
        Process          Burst Time      Waiting Time    Turnaround Time
                2                3                0                3
                3                3                3                6
                1                24               6                30
Average waiting time: 3.000000
Average turnaround time: 13.000000
```

Program 8: Write a program for SJF (preemptive) algorithm.

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

int main(){
    int i, j, n;
    float sumw = 0, sumt = 0;
    int w[10], b[10], a[10], r[10], t[10];
    float avgw, avgt;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    int time, remain = n;
    b[9] = 9999;
    for(i=0; i<n; i++){
        printf("Enter the burst time for process %d: ", (i+1));
        scanf("%d", &b[i]);
        printf("Enter the arrival time for process %d: ", (i+1));
        scanf("%d", &a[i]);
        r[i] = b[i];
    }

    printf("\tProcess \t Burst Time \t Arrival Time \t Waiting Time \t
Turnaround Time\n");
    for(time = 0; remain != 0; time++){
        int s = 9;
        for(i=0; i<n; i++){
            if(a[i] <= time && b[i] <= b[s] && r[i] > 0){
                s=i;
            }
        }
        r[s]--;
        if(r[s] == 0){
            remain--;
            w[s] = time + 1 - a[s] - b[s];
            t[s] = time + 1 -a[s];
            printf("\t\t %d \t\t %d \t\t %d \t\t %d \t\t %d\n", (s+1),
b[s], a[s], w[s], t[s]);
            sumw += w[s];
            sumt += t[s];
        }
    }
    avgw = sumw/n;
    avgt = sumt/n;
    printf("Average waiting time: %f\n", avgw);
    printf("Average turnaround time: %f\n", avgt);
}
```

```
PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab> cd "c:\Users\sriha\
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc sjfp.c -o sjfp } ; if ($?) { .\sjfp }
Enter the number of processes: 3
Enter the burst time for process 1: 24
Enter the arrival time for process 1: 0
Enter the burst time for process 2: 3
Enter the arrival time for process 2: 2
Enter the burst time for process 3: 3
Enter the arrival time for process 3: 1
        Process         Burst Time      Arrival Time    Waiting Time    Turnaround Time
                3               3               1               0               3
                2               3               2               2               5
                1               24              0               6               30
Average waiting time: 2.666667
Average turnaround time: 12.666667
```

Program 9: Write a program for Priority (non-preemptive) algorithm.

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

int main(){
    int i, n, temp;
    float sumw = 0, sumt = 0;
    int w[10], b[10], t[10], p[10];
    float avgw, avgt;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    for(i=1; i<=n; i++){
        printf("Enter the priority for process %d: ", i);
        scanf("%d", &temp);
        printf("Enter the burst time for process %d: ", i);
        scanf("%d", &b[temp]);
        p[temp] = i;
    }
    w[1]=0;
    printf("\tProcess \t Burst Time \t Priority \t Waiting Time \t
Turnaround Time\n");
    for(i=1; i<=n; i++){
        t[i] = b[i] + w[i];
        printf("\t\t %d \t\t %d \t\t %d \t\t %d \t\t %d\n", p[i], b[i], i,
w[i], t[i]);
        w[i+1] = w[i] + b[i];
        sumw += w[i];
        sumt += t[i];
    }
    avgw = sumw/n;
    avgt = sumt/n;
    printf("Average waiting time: %f\n", avgw);
    printf("Average turnaround time: %f\n", avgt);
}
```

```
PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab> cd "c:\Users\srih
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc pnp.c -o pnp } ; if ($?) { .\pnp }
Enter the number of processes: 5
Enter the priority for process 1: 3
Enter the burst time for process 1: 10
Enter the priority for process 2: 1
Enter the burst time for process 2: 1
Enter the priority for process 3: 4
Enter the burst time for process 3: 2
Enter the priority for process 4: 5
Enter the burst time for process 4: 1
Enter the priority for process 5: 2
Enter the burst time for process 5: 5
        Process         Burst Time      Priority        Waiting Time    Turnaround Time
                2               1               1               0               1
                5               5               2               1               6
                1               10              3               6               16
                3               2               4               16              18
                4               1               5               18              19
Average waiting time: 8.200000
Average turnaround time: 12.000000
```

Program 10: Write a program for Priority (preemptive) algorithm.

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

int main(){
    int i, j, n;
    float sumw = 0, sumt = 0;
    int w[10], b[10], a[10], p[10], r[10], t[10];
    float avgw, avgt;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    int time, remain = n;
    b[9] = 9999;
    for(i=1; i<=n; i++){
        printf("Enter the priority for process %d: ", i);
        scanf("%d", &p[i]);
        printf("Enter the burst time for process %d: ", i);
        scanf("%d", &b[i]);
        printf("Enter the arrival time for process %d: ", i);
        scanf("%d", &a[i]);
        r[i] = b[i];
    }

    printf("\tProcess \t Burst Time \t Arrival Time \t Priority \t Waiting Time \t Turnaround Time\n");
    for(time = 0; remain != 0; time++){
        int s = 9;
        for(i=1; i<=n; i++){
            if(a[i] <= time && p[i] <= p[s] && r[i] > 0){
                s=i;
            }
        }
        r[s]--;
        if(r[s] == 0){
            remain--;
            w[s] = time + 1 - a[s] - b[s];
            t[s] = time + 1 -a[s];
            printf("\t\t %d \t\t %d \t\t %d \t\t %d \t\t %d \t\t %d\n", (s+1), b[s], a[s], p[s], w[s], t[s]);
            sumw += w[s];
            sumt += t[s];
        }
    }
    avgw = sumw/n;
    avgt = sumt/n;
    printf("Average waiting time: %f\n", avgw);
    printf("Average turnaround time: %f\n", avgt);
}
```

```
PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab> cd "c:\Users\sriha\OneDrive\Doc
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc pp.c -o pp } ; if ($?) { .\pp }
Enter the number of processes: 5
Enter the priority for process 1: 3
Enter the burst time for process 1: 10
Enter the arrival time for process 1: 0
Enter the priority for process 2: 1
Enter the burst time for process 2: 1
Enter the arrival time for process 2: 1
Enter the priority for process 3: 4
Enter the burst time for process 3: 2
Enter the arrival time for process 3: 2
Enter the priority for process 4: 5
Enter the burst time for process 4: 1
Enter the arrival time for process 4: 3
Enter the priority for process 5: 2
Enter the burst time for process 5: 5
Enter the arrival time for process 5: 4
        Process         Burst Time      Arrival Time    Priority        Waiting Time    Turnaround Time
                3               1               1               1               0               1
                6               5               4               2               0               5
                2               10              0               3               6               16
                4               2               2               4               14              16
                5               1               3               5               15              16
Average waiting time: 7.000000
Average turnaround time: 10.800000
```

Program 11: Write a program for Round Robin scheduling algorithm.

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

int main(){
    int i, n, flag = 0, tq, time, remain;
    float sumw = 0, sumt = 0;
    int w[10], b[10], t[10], p[10], r[10];
    float avgw, avgt;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    remain = n;
    printf("Enter time slice: ");
    scanf("%d", &tq);
    for(i=0; i<n; i++){
        printf("Enter the burst time for process %d: ", (i+1));
        scanf("%d", &b[i]);
        r[i] = b[i];
        p[i] = i+1;
    }
    printf("\tProcess \t Burst Time \t Waiting Time \t Turnaround Time\n");
    for(time=0, i= 0; remain !=0;){
        if(r[i] <= tq && r[i] > 0){
            time += r[i];
            r[i] = 0;
            flag = 1;
        }else if(r[i] > 0){
            r[i] -= tq;
            time += tq;
        }
        if(r[i] == 0 && flag == 1){
            remain --;
            w[i] = time - b[i];
            t[i] = time;
            printf("\t\t %d \t\t %d \t\t %d \t\t %d\n", p[i], b[i], w[i],
t[i]);
            sumw += w[i];
            sumt += t[i];
            flag = 0;
        }
        if(i == n-1){
            i=0;
        }else{
            i++;
        }
    }
    avgw = sumw/n;
    avgt = sumt/n;
    printf("Average waiting time: %f\n", avgw);
    printf("Average turnaround time: %f\n", avgt);
}
```

```
cd "c:\Users\
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc rr.c -o rr } ; if ($?) { .\rr }
Enter the number of processes: 3
Enter time slice: 1
Enter the burst time for process 1: 24
Enter the burst time for process 2: 3
Enter the burst time for process 3: 3
        Process          Burst Time      Waiting Time    Turnaround Time
                2                3               5               8
                3                3               6               9
                1                24              6               30
Average waiting time: 5.666667
Average turnaround time: 15.666667
```

Program 12: Write a program to demonstrate:

A. First Fit

```c
#include<stdio.h>

int main(){
    int i, j, n, flag;
    printf("Enter the number of files: ");
    scanf("%d", &n);
    int files[100], holes[100];
    printf("Enter the size of each file.\n");
    for(i=0; i<n; i++){
        scanf("%d", &files[i]);
    }
    printf("Enter the size of each hole.\n");
    for(i=0; i<n; i++){
        scanf("%d", &holes[i]);
    }
    printf("File No. \t File Size \t Hole Size \t Memory Wasted \n");
    for(i=0; i<n; i++){
        flag = 0;
        for(j=0; j<n; j++){
            if(files[i] <= holes[j]){
                flag = 1;
                printf("%d \t\t %d \t\t %d \t\t %d \n", i+1, files[i],
holes[j], holes[j] - files[i]);
                holes[j] = 0;
                break;
            }
        }
        if(flag == 0){
            printf("%d \t\t %d \t\t Unable to allocate memory.\n", i+1,
files[i]);
        }
    }
}
```

```
Number of page faults is: 15
PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab> cd "c:
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc firstfit.c -o firstfit }
Enter the number of files: 5
Enter the size of each file.
6 29 2 256 1
Enter the size of each hole.
500 250 6 7 199
File No.        File Size       Hole Size       Memory Wasted
1               6               500             494
2               29              250             221
3               2               6               4
4               256             Unable to allocate memory.
5               1               7               6
```

B. Best Fit

```
#include<stdio.h>

int main(){
    int i, j, n, temp, flag;
    printf("Enter the number of files: ");
    scanf("%d", &n);
    int files[100], holes[100];
    printf("Enter the size of each file.\n");
    for(i=0; i<n; i++){
        scanf("%d", &files[i]);
    }
    printf("Enter the size of each hole.\n");
    for(i=0; i<n; i++){
        scanf("%d", &holes[i]);
    }
    for(i=0; i<n; i++){
        for(j=i+1; j<n; j++){
            if(holes[i] > holes[j]){
                temp = holes[j];
                holes[j] = holes[i];
                holes[i] = temp;
            }
        }
    }
    printf("File No. \t File Size \t Hole Size \t Memory Wasted \n");
    for(i=0; i<n; i++){
        flag = 0;
        for(j=0; j<n; j++){
            if(files[i] <= holes[j]){
                flag = 1;
                printf("%d \t\t %d \t\t %d \t\t %d \n", i+1, files[i],
holes[j], holes[j] - files[i]);
                holes[j] = 0;
                break;
            }
        }
        if(flag == 0){
            printf("%d \t\t %d \t\t Unable to allocate memory.\n", i+1,
files[i]);
        }
    }
}
```

```
PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab>
ty\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc bestfit.c -o bestfit } ; if ($?) {
Enter the number of files: 5
Enter the size of each file.
6 29 2 256 1
Enter the size of each hole.
500 250 6 7 199
File No.        File Size       Hole Size       Memory Wasted
1               6               6               0
2               29              199             170
3               2               7               5
4               256             500             244
5               1               250             249
```

C. Worst fit

```c
#include<stdio.h>

int main(){
    int i, j, n, temp, flag;
    printf("Enter the number of files: ");
    scanf("%d", &n);
    int files[100], holes[100];
    printf("Enter the size of each file.\n");
    for(i=0; i<n; i++){
        scanf("%d", &files[i]);
    }
    printf("Enter the size of each hole.\n");
    for(i=0; i<n; i++){
        scanf("%d", &holes[i]);
    }
    for(i=0; i<n; i++){
        for(j=i+1; j<n; j++){
            if(holes[i] < holes[j]){
                temp = holes[j];
                holes[j] = holes[i];
                holes[i] = temp;
            }
        }
    }
    printf("File No. \t File Size \t Hole Size \t Memory Wasted \n");
    for(i=0; i<n; i++){
        flag = 0;
        for(j=0; j<n; j++){
            if(files[i] <= holes[j]){
                flag = 1;
                printf("%d \t\t %d \t\t %d \t\t %d \n", i+1, files[i],
holes[j], holes[j] - files[i]);
                holes[j] = 0;
                break;
            }
        }
        if(flag == 0){
            printf("%d \t\t %d \t\t Unable to allocate memory.\n", i+1,
files[i]);
        }
    }
```

```
40}
```

```
PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab> cd "c:
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc worstfit.c -o worstfit }
Enter the number of files: 5
Enter the size of each file.
6 29 2 256 1
Enter the size of each hole.
500 250 6 7 199
File No.        File Size       Hole Size       Memory Wasted
1               6               500             494
2               29              250             221
3               2               199             197
4               256             Unable to allocate memory.
5               1               7               6
```

Program 13: Write a program for FIFO page replacement algorithm.

```c
#include<stdio.h>

int main(){
    int i, j, n, m, point = 0, flag, fault = 0;
    printf("Enter the number of pages:");
    scanf("%d", &n);
    int pages[100];
    printf("Enter the reference string: ");
    for (i=0; i<n; i++){
        scanf("%d", &pages[i]);
    }
    printf("Enter the number of frames.");
    scanf("%d", &m);
    int frames[100], max[100];
    for(j=0; j<m; j++){
        frames[j] = -1;
        max[m] = 9999;
    }
    for(i=0; i<n; i++){
        flag = 0;
        for(j=0; j<m; j++){
            if(frames[j] == pages[i]){
                flag = 1;
            }
        }
        if(flag == 0){
            frames[point] = pages[i];
            if(point == m-1){
                point = 0;
            }else{
                point++;
            }
        }
        if(flag == 0){
            fault++;
            printf("Page Fault: ");
            for(j=0; j<m; j++){
                printf("%d ", frames[j]);
            }
            printf("\n");
        }
    }
    printf("Number of page faults is: %d ", fault);
}
```

```
Average turnaround time: 27.000000
PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc fifo.c -o fifo } ;
Enter the number of pages:20
Enter the reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Enter the number of frames.3
Page Fault: 7 -1 -1
Page Fault: 7 0 -1
Page Fault: 7 0 1
Page Fault: 2 0 1
Page Fault: 2 3 1
Page Fault: 2 3 0
Page Fault: 4 3 0
Page Fault: 4 2 0
Page Fault: 4 2 3
Page Fault: 0 2 3
Page Fault: 0 1 3
Page Fault: 0 1 2
Page Fault: 7 1 2
Page Fault: 7 0 2
Page Fault: 7 0 1
Number of page faults is: 15
```

Program 14: Write a program for Optimal page replacement algorithm.

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

int main(){
    int i, j, k, n, m, repl, flag, point, fault = 0;
    printf("Enter the number of pages:");
    scanf("%d", &n);
    int pages[100];
    printf("Enter the reference string: ");
    for (i=0; i<n; i++){
        scanf("%d", &pages[i]);
    }
    printf("Enter the number of frames.");
    scanf("%d", &m);
    int frames[100], max[100];
    for(j=0; j<m; j++){
        frames[j] = -1;
        max[m] = 9999;
    }
    for(i=0; i<n; i++){
        flag = 0;
        for(j=0; j<m; j++){
            if(frames[j] == pages[i]){
                flag = 1;
            }
        }
        if(flag == 0 && i >= m){
            for(j=0; j<m; j++){
                point = 0;
                for(k = i+1; k<n; k++){
                    if(pages[k] != frames[j]){
                        point++;
                    }else{
                        max[j] = point;
                        break;
                    }
                }
            }
            repl = 0;
            for(j=0; j<m; j++){
                if(max[repl] < max[j]){
                    repl = j;
                }
            }
            frames[repl] = pages[i];
        }
        if(flag == 0){
            if(i < m){
                frames[i] = pages[i];
            }
            fault++;
            printf("Page Fault: ");
            for(j=0; j<m; j++){
                max[j] = 9999;
```

```
55                 printf("%d ", frames[j]);
56             }
57             printf("\n");
58         }
59     }
60     printf("Number of page faults is: %d ", fault);
61}
```

```
PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab> cd "
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc optimal.c -o optimal }
Enter the number of pages:20
Enter the reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Enter the number of frames.3
Page Fault: 7 -1 -1
Page Fault: 7 0 -1
Page Fault: 7 0 1
Page Fault: 2 0 1
Page Fault: 2 0 3
Page Fault: 2 4 3
Page Fault: 2 0 3
Page Fault: 2 0 1
Page Fault: 7 0 1
Number of page faults is: 9
```

Program 15: Write a program for LRU page replacement algorithm.

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

int main(){
    int i, j, k, n, m, repl, flag, point, fault = 0;
    printf("Enter the number of pages:");
    scanf("%d", &n);
    int pages[100];
    printf("Enter the reference string: ");
    for (i=0; i<n; i++){
        scanf("%d", &pages[i]);
    }
    printf("Enter the number of frames.");
    scanf("%d", &m);
    int frames[100], max[100];
    for(j=0; j<m; j++){
        frames[j] = -1;
        max[m] = 9999;
    }
    for(i=0; i<n; i++){
        flag = 0;
        for(j=0; j<m; j++){
            if(frames[j] == pages[i]){
                flag = 1;
            }
        }
        if(flag == 0 && i >= m){
            for(j=0; j<m; j++){
                point = 0;
                for(k = i-1; k>=0; k--){
                    if(pages[k] != frames[j]){
                        point++;
                    }else{
                        max[j] = point;
                        break;
                    }
                }
            }
            repl = 0;
            for(j=0; j<m; j++){
                if(max[j] > max[repl]){
                    repl = j;
                }
            }
            frames[repl] = pages[i];
        }
        if(flag == 0){
            if(i < m){
                frames[i] = pages[i];
            }
            fault++;
            printf("Page Fault: ");
            for(j=0; j<m; j++){
                max[j] = 9999;
```

```
55                 printf("%d ", frames[j]);
56             }
57             printf("\n");
58         }
59     }
60     printf("Number of page faults is: %d ", fault);
61}
```

```
PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\L
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc lru.c -o lru }
Enter the number of pages:20
Enter the reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Enter the number of frames.3
Page Fault: 7 -1 -1
Page Fault: 7 0 -1
Page Fault: 7 0 1
Page Fault: 2 0 1
Page Fault: 2 0 3
Page Fault: 4 0 3
Page Fault: 4 0 2
Page Fault: 4 3 2
Page Fault: 0 3 2
Page Fault: 1 3 2
Page Fault: 1 0 2
Page Fault: 1 0 7
Number of page faults is: 12
```

Program 16: Write a program to demonstrate producer-consumer problem.

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

sem_t empty, full, mutex;
char buf[10];

void *producer(void* arg){
    int i;
    for(i=0; i<10; i++){
        sem_wait(&empty);
        sem_wait(&mutex);
        buf[i] = i;
        printf("Item produced is %d\n", buf[i]);
        sem_post(&mutex);
        sem_post(&full);
        sleep(1);
    }
    pthread_exit("Producer\n");
}

void *consumer(void* arg){
    int j;
    for(j=0; j<10; j++){
        sem_wait(&full);
        sem_wait(&mutex);
        j = buf[j];
        printf("Item consumed is %d\n", buf[j]);
        sem_post(&mutex);
        sem_post(&empty);
        sleep(5);
    }
    pthread_exit("Consumer\n");
}

int main(){
    pthread_t pid1,pid2;
    sem_init(&empty, 0, 10);
    sem_init(&full, 0, 0);
    sem_init(&mutex, 1, 1);
    void *status;
    pthread_create(&pid1, NULL, producer, NULL);
    pthread_create(&pid2, NULL, consumer, NULL);
    pthread_join(pid1, &status);
    printf("The exited status of 1st is %s\n", (char*)status);
    pthread_join(pid2,&status);
    printf("The exited status %s\n", (char*)status);
    return 0;
}
```

```
PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc pc.c -o pc }
Item produced is 0
Item consumed is 0
Item produced is 1
Item produced is 2
Item produced is 3
Item produced is 4
Item consumed is 1
Item produced is 5
Item produced is 6
Item produced is 7
Item produced is 8
Item produced is 9
Item consumed is 2
The exited status of 1st is Producer

Item consumed is 3
Item consumed is 4
Item consumed is 5
Item consumed is 6
Item consumed is 7
Item consumed is 8
Item consumed is 9
The exited status Consumer
```

Program 17: Write a program to demonstrate readers-writers problem.

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

int data=0, rdcnt=0;

sem_t mutex, writeblock;

void * reader(void * no){
    printf("\n\tReader %d is executing ", (int)no);
    sem_wait(&mutex);
    printf("\n\tWait to mutex by %d reader", (int)no);
    rdcnt++;
    if(rdcnt==1){
        sem_wait(&writeblock);
        printf("\n\tWait to writerblock by %d reader", (int)no);
    }
    printf("\n\t***Reader %d read data = %d ", (int)no, data);
    if(rdcnt==1){
        sem_post(&writeblock);
        printf("\n\tSignal to writerblock by %d reader", (int)no);
    }
    sem_post(&mutex);
    printf("\n\tSignal to mutex by %d reader\n", (int)no);
}

void * writer(void * no){
    printf("\n\tWriter %d is executing ", (int)no);
    sem_wait(&writeblock);
    printf("\n\tWait to writerblock by %d writer", (int)no);
    data+=5;
    printf("\n\t***Writer %d write data = %d ", (int)no, data);
    sem_post(&writeblock);
    printf("\n\tSignal to writer by %d writer\n", (int)no);
}

int main(){
    int no, i, ir=0, iw=0, ch;
    sem_init(&mutex, 0, 1);
    sem_init(&writeblock, 0, 1);
    printf("\nEnter no of readers and writers to create: ");
    scanf("%d", &no);
    pthread_t r[no],w[no];
    do{
        printf("\n\t1. Reader\n\t2. Writer\n\t3. Terminate\n\tYour choice:
");
        scanf("%d",&ch);
        switch(ch){
            case 1: pthread_create(&r[ir], NULL, reader, (void *)ir);
                    pthread_join(r[ir++], NULL);
                    break;
            case 2: pthread_create(&w[iw], NULL, writer, (void *)iw);
                    pthread_join(w[iw++], NULL);
```

```
                    break;
            }
        }while(ch!=3);

        sem_destroy(&mutex);
        sem_destroy(&writeblock);
}
```

```
Enter no of readers and writers to create: 2

        1. Reader
        2. Writer
        3. Terminate
        Your choice: 1

        Reader 0 is executing
        Wait to mutex by 0 reader
        Wait to writerblock by 0 reader
        ***Reader 0 read data = 0
        Signal to writerblock by 0 reader
        Signal to mutex by 0 reader

        1. Reader
        2. Writer
        3. Terminate
        Your choice: 2

        Writer 0 is executing
        Wait to writerblock by 0 writer
        ***Writer 0 write data = 5
        Signal to writer by 0 writer

        1. Reader
        2. Writer
        3. Terminate
        Your choice: 3
```

Program 18: Write a program to demonstrate dining-philosophers problem.

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

#define N 5
#define THINKING 0
#define HUNGRY 1
#define EATING 2
#define LEFT (ph_num+4)%N
#define RIGHT (ph_num+1)%N

sem_t mutex;
sem_t S[N];

void * philosopher(void *num);
void take_fork(int);
void put_fork(int);
void test(int);

int state[N];
int phil_num[N] = {0, 1, 2, 3, 4};

int main(){
    int i;
    pthread_t thread_id[N];
    sem_init(&mutex, 0, 1);
    for(i=0; i<N; i++){
        pthread_create(&thread_id[i], NULL, philosopher, &phil_num[i]);
        printf("Philosopher %d is thinking.\n", i+1);
    }
    for(i=0; i<N; i++){
        pthread_join(thread_id[i], NULL);
    }
}

void *philosopher(void *num){
    while(1){
        int *i = num;
        sleep(1);
        take_fork(*i);
        sleep(0);
        put_fork(*i);
    }
}

void take_fork(int ph_num){
    sem_wait(&mutex);
    state[ph_num] = HUNGRY;
    printf("Philosopher %d is hungry.\n", ph_num+1);
    test(ph_num);
    sem_post(&mutex);
    sem_wait(&S[ph_num]);
    sleep(1);
```

```
54}
55
56void test(int ph_num){
57    if(state[ph_num] = HUNGRY && state[LEFT] != EATING && state[RIGHT] !=
EATING){
58        state[ph_num] = EATING;
59        sleep(2);
60        printf("Philosopher %d takes chopsticks %d and %d.\n", ph_num+1,
LEFT+1, ph_num+1);
61        printf("Philosopher %d is eating.\n", ph_num+1);
62        sem_post(&S[ph_num]);
63    }
64}
65
66void put_fork(int ph_num){
67    sem_wait(&mutex);
68    state[ph_num] = THINKING;
69    printf("Philosopher %d puts chopstick %d and %d down.\n", ph_num+1,
LEFT+1, ph_num+1);
70    printf("Philosopher %d is thinking.\n", ph_num+1);
71    test(LEFT);
72    test(RIGHT);
73    sem_post(&mutex);
74}
```

```
PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc dp.c -o dp }
dp.c: In function 'philosopher':
dp.c:39:9: warning: implicit declaration of function 'sleep' [-Wimplicit
   39 |         sleep(1);
      |         ^~~~~
Philosopher 1 is thinking.
Philosopher 2 is thinking.
Philosopher 3 is thinking.
Philosopher 4 is thinking.
Philosopher 5 is thinking.
Philosopher 1 is hungry.
Philosopher 1 takes chopsticks 5 and 1.
Philosopher 1 is eating.
Philosopher 4 is hungry.
Philosopher 4 takes chopsticks 3 and 4.
Philosopher 4 is eating.
Philosopher 2 is hungry.
Philosopher 5 is hungry.
Philosopher 3 is hungry.
Philosopher 1 puts chopstick 5 and 1 down.
Philosopher 1 is thinking.
Philosopher 2 takes chopsticks 1 and 2.
Philosopher 2 is eating.
Philosopher 2 puts chopstick 1 and 2 down.
Philosopher 2 is thinking.
Philosopher 1 takes chopsticks 5 and 1.
Philosopher 1 is eating.
Philosopher 4 puts chopstick 3 and 4 down.
Philosopher 4 is thinking.
Philosopher 3 takes chopsticks 2 and 3.
Philosopher 3 is eating.
Philosopher 5 puts chopstick 4 and 5 down.
Philosopher 5 is thinking.
```

Program 19: Write a program to demonstrate file reading and writing.

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#define BUFFER_SIZE 1024
typedef struct {
    FILE *input_file;
    FILE *output_file;
} ThreadData;
void *copyFile(void *arg) {
    ThreadData *data = (ThreadData *)arg;
    char buffer[BUFFER_SIZE];
    size_t bytesRead;
    while ((bytesRead = fread(buffer, 1, BUFFER_SIZE, data->input_file)) > 0) {
        fwrite(buffer, 1, bytesRead, data->output_file);
    }
    fclose(data->input_file);
    fclose(data->output_file);
    return NULL;
}
int main() {
    pthread_t thread;
    ThreadData data;
    data.input_file = fopen("input.txt", "rb");
    if (data.input_file == NULL) {
        perror("Error opening input file");
        exit(EXIT_FAILURE);
    }
    data.output_file = fopen("output.txt", "wb");
    if (data.output_file == NULL) {
        perror("Error opening output file");
        fclose(data.input_file);
        exit(EXIT_FAILURE);
    }
    if (pthread_create(&thread, NULL, copyFile, &data) != 0) {
        perror("Error creating thread");
        fclose(data.input_file);
        fclose(data.output_file);
        exit(EXIT_FAILURE);
    }
    if (pthread_join(thread, NULL) != 0) {
        perror("Error joining thread");
        fclose(data.input_file);
        fclose(data.output_file);
        exit(EXIT_FAILURE);
    }
    printf("File copied successfully.\n");
    return 0;
}
```

```
indi@Harini:~$ ./a.out
File copied successfully.
```

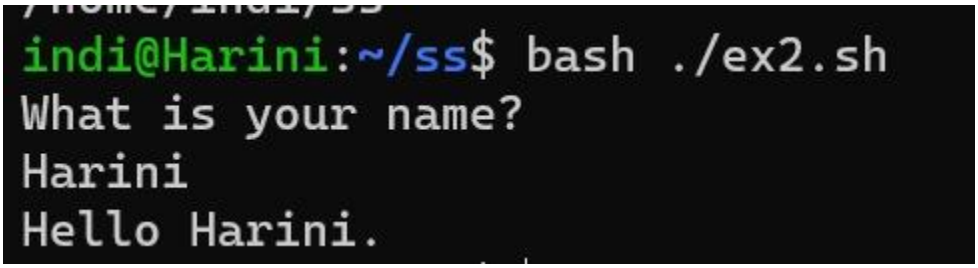
SHELL SCRIPT:

1. Write a shell script to show basic Linux commands.

```
# SS1
ls
who
pwd
```

2. Write an interactive shell script that takes one input.

```
# SS2
echo What is your name?
read name
echo Hello $name.
```



3. Write an interactive shell script that takes multiple user inputs.

```
# SS3
echo "Please enter your surname"
echo "followed by your first name: "
read name1 name2
echo "Welcome to CSE Dept, UCEOU, $name2 $name1"
```

```
indi@Harini:~/ss$ bash ./ex3.sh
Please enter your surname
followed by your first name:
Margapuri Sriharini
Welcome to CSE Dept, UCEOU, Sriharini Margapuri
```

4. Write a shell script that takes to file names as input and copies first file into the second file.

```
# SS4
echo “Please Enter source file name :”
read source
echo “Enter the target file name :”
read target
cp $source $target
echo file $source is copied into the $target
```

```
indi@Harini:~/ss$ bash ./ex4.sh
Please Enter source file name :
source.txt
Enter the target file name :
target.txt
file source.txt is copied into the target.txt
```

5. **Write shell script which will accept 5 numbers as parameters and display their sum. Also display the contents of the different variables in the script.**

```
# SS5

echo the parameters passed are : $1, $2, $3, $4, $5

echo the name of the script is : $0

echo the number of parameters passed are : $#

sum=`expr $1 + $2 + $3 + $4 + $5` echo

The sum is : $sum
```

```
indi@Harini:~/ss$ bash ./ex5.sh 2 4 6 8 10
the parameters passed are : 2, 4, 6, 8, 10
the name of the script is : ./ex5.sh
the number of parameters passed are : 5
The sum is : 30
```

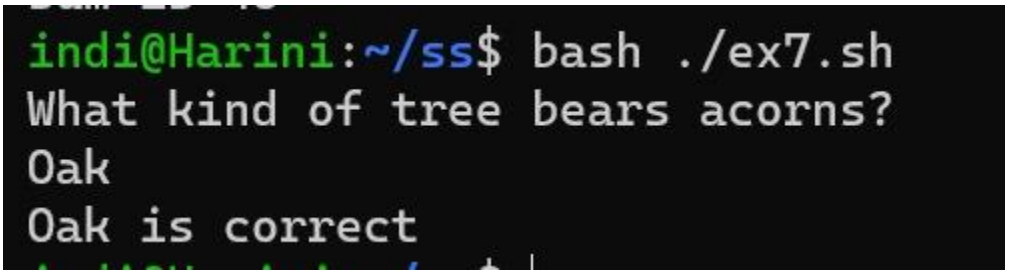
6. **Write a shell script which will accept different numbers and finds their sum. The number of parameters can vary.**

```
#SS6

sum=0

while [ $# -gt 0 ]

do

sum=`expr $sum + $1`

shift

done

echo sum is $sum
```

```
indi@Harini:~/ss$ bash ./ex6.sh 1 2 3 4 5 6 7 8 9 0
sum is 45
```

7. **Write a shell script that takes user's answers and returns whether it is correct using case.**

```
#SS7
echo What kind of tree bears acorns?
read responce
case $responce in
[Oo][Aa][Kk]) echo $responce is correct ;;
*) echo Sorry, responce is wrong
 esac
```



8. **Write a shell script that takes user's answers and keeps asking for inputs until user gets the answer right using while.**

```
#SS8
echo What is the Capital of Saudi Arabia? R
ead answer
while test $answer != Riyadh
do echo No, Wrong please try again.
read answer
done
echo This is correct.
```

```
What is the Capital of Saudi Arabia?
Dubai
No, Wrong please try again.
Riyadh
This is correct
indi@Harini:~/ss$
```

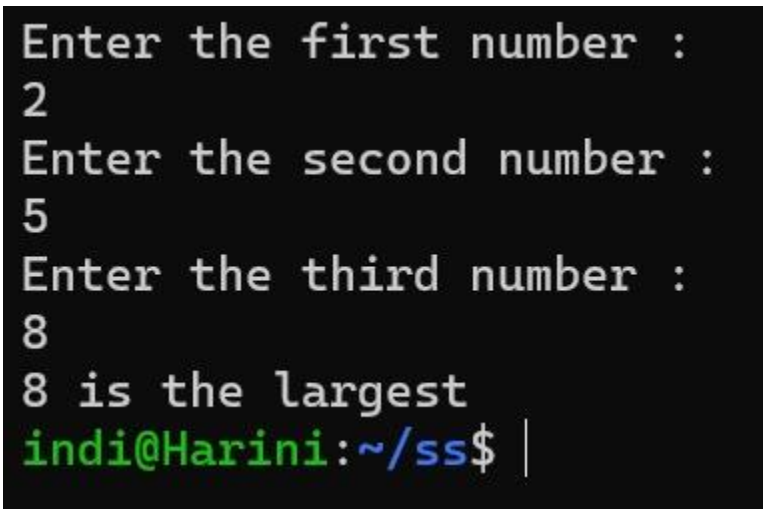
9. Write a shell script that asks for user login name using until.

```
#SS9
echo "Please Enter the user login name: "
read login_name
until who | grep $login_name
do
sleep 30
done
echo The user $login_name has logged in
```

```
Please Enter the user login name:
indi
indi
The user indi has logged in
indi@Harini:~/ss$
```

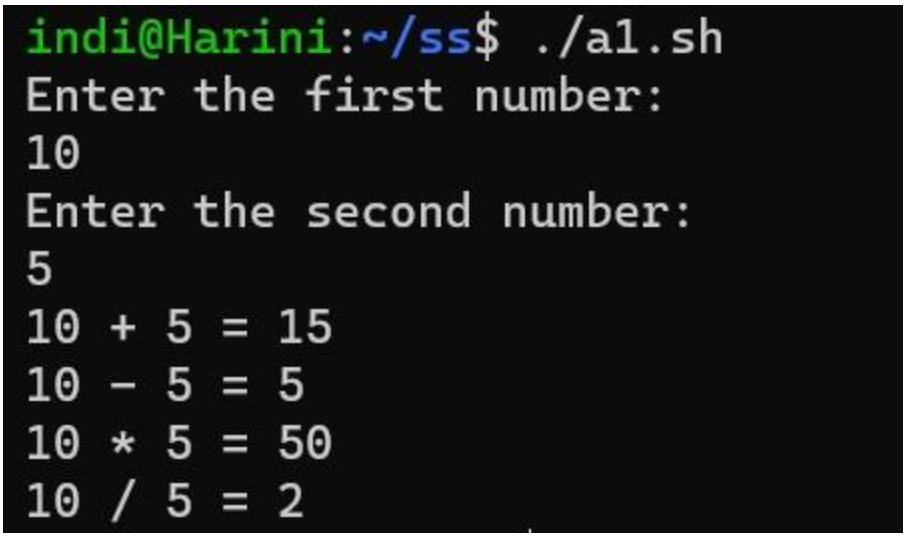
10. Write a shell script that takes 3 numbers from the user and determines the largest of those 3 numbers using if.

```
#SS10
echo "Enter the first number :"
read num1
echo "Enter the second number :"
read num2
echo "Enter the third number :"
read num3
if test $num1 -gt $num2
then
      if test $num1 -gt $num3
      then
            echo $num1 is the largest
      else
            echo $num3 is the largest
      fi
else
      if test $num2 -gt $num3
      then
            echo $num2 is largest
      else
            echo $num3 is the largest
       fi
fi
```



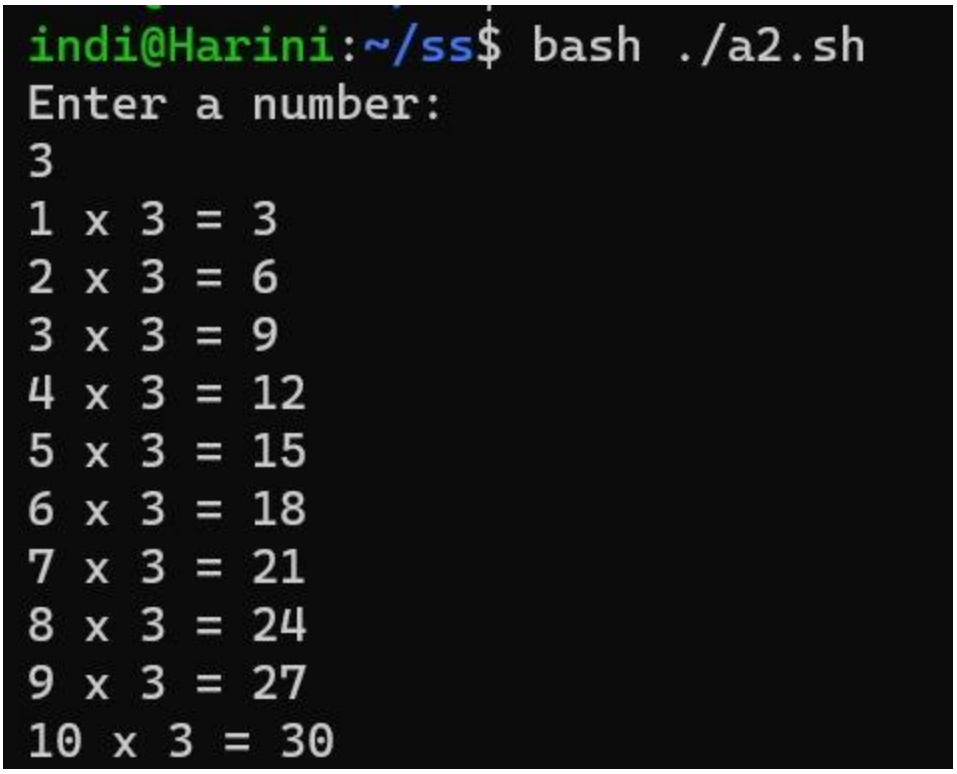
11. Write a shell script to demonstrate arithmetic operators.

```
echo Enter the first number:
read num1
echo Enter the second number:
read num2

sum=expr $num1 + $num2
dif=expr $num1 - $num2
pro=expr $num1 \* $num2
quo=expr $num1 / $num2

echo $num1 + $num2 = $sum
echo $num1 - $num2 = $dif
echo $num1 \* $num2 = $pro
echo $num1 / $num2 = $quo
```



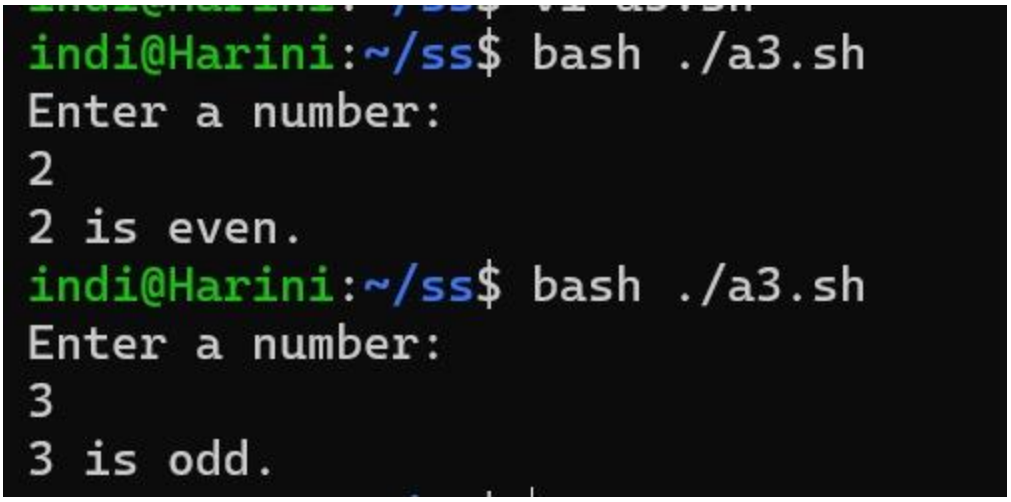
12. Write a shell script to demonstrate do-while.

```
echo Enter a number:
read num
i=1
while [ $i -le 10 ]
do
        pro=expr $i \* $num
        echo $i x $num = $pro
        i=expr $i + 1
done
```



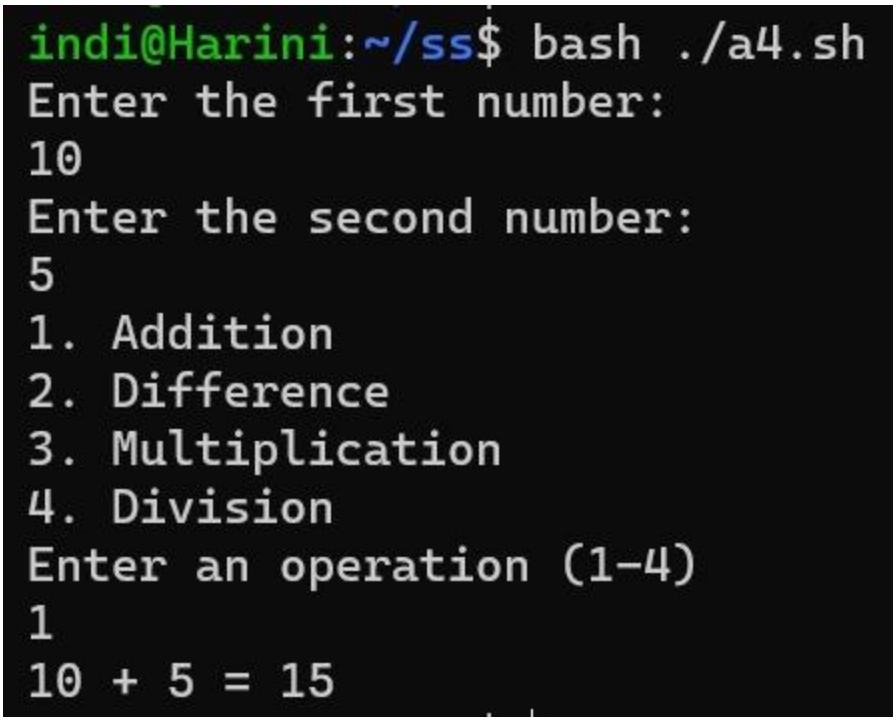
13. Write a shell script to demonstrate if-else.

```
echo Enter a number:
read num
rem=expr $num % 2
if [ $rem -eq 0 ]
then
        echo $num is even.
else
        echo $num is odd.
fi
```



14. Write a shell script to demonstrate case.

```
echo Enter the first number:
read num1
echo Enter the second number:
read num2

echo 1. Addition
echo 2. Difference
echo 3. Multiplication
echo 4. Division

echo "Enter an operation (1-4)"
read op

case $op in
1) sum=expr $num1 + $num2
        echo $num1 + $num2 = $sum;;
2) dif=expr $num1 - $num2
        echo $num1 - $num2 = $dif;;
3) pro=expr $num1 \* $num2
        echo $num1 \* $num2 = $pro;;
4) quo=expr $num1 / $num2
        echo $num1 / $num2 = $quo;;
*) echo Terminating...;;
esac
```



15. Write a shell script to demonstrate logical operators.

```
read -p 'Enter a : ' a
read -p 'Enter b : ' b

if(($a == "true" && $b == "true" ))
then
        echo Both are true.
else
        echo Both are not true.
fi

if(($a == "true" || $b == "true" ))
then
        echo Atleast one of them is true.
else
        echo None of them is true.
fi

if(( ! $a == "true" ))
then
        echo "a" was initially false.
else
        echo "a" was initially true.
fi
```

```
indi@Harini:~/ss$ bash ./a5.sh
Enter a : true
Enter b : false
Both are true.
Atleast one of them is true.
a was initially true.
```