ST. JOSEPH UNIVERSITY IN TANZANIA

St. Joseph College of Engineering and Technology
Department of Computer Science and Engineering
Operating Systems Laboratory 055CS58
By Robert Innocent Karamagi

List of Experiments

(Implement the following on LINUX or other Unix like platform. Use C for high level language implementation)

1(a) Implementation of process management using the following system calls of UNIX operating system: fork, exec, getpid, exit, wait, close.

1(b) Implementation of directory management using the following system calls of UNIX operating system: opendir, readdir.

2 Write a program using the I/O system calls of UNIX operating system (open,read, write, etc)

3 Write a programs to simulate UNIX commands like ls, grep, etc.

4(a) Write a program for implementing the First Come First Served Scheduling algorithm

4(b) Write a program for simulation of Shortest Job First Scheduling algorithm

5(a) Write a program for implementing the Round Robin Scheduling algorithm

5(b) Write a program for implementing the Priority scheduling algorithm

6 Developing Application using Inter Process communication (using shared memory, pipes or message queues)

7 Implement the Producer – Consumer problem using semaphores (using UNIX system calls).

8 Implement first fit algorithm for memory management

9(a) Implement Best fit algorithm for memory management

9(b) Implement worst fit algorithm for memory management

10 Implement Contiguous file allocation technique.

Experiment 1a

/*UNIX System Calls: fork,exec,getpid,exit,wait,close,stat*/

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

int main(int argc, char*argv[])
{
        int pid;
        char s[100];
        pid=fork();
        argv[2]="date";
        if(pid<0)
        printf("Error.");

        else if(pid>0)
        {
                wait(NULL);
                printf("\nParent process\n");
                printf("\nParent process ID : %d\n\n",getpid());
                execlp("stat","stat","ex1a.c",NULL);
                exit(0);
        }

        else
        {
                printf("\nChild process\n");
                printf("\nChild process parent ID : %d\n",getppid());
                printf("\nChild process ID : %d\n\n",getpid());
                execvp(argv[2],argv);
                close(pid);

        }
        return 0;
}
```

Output

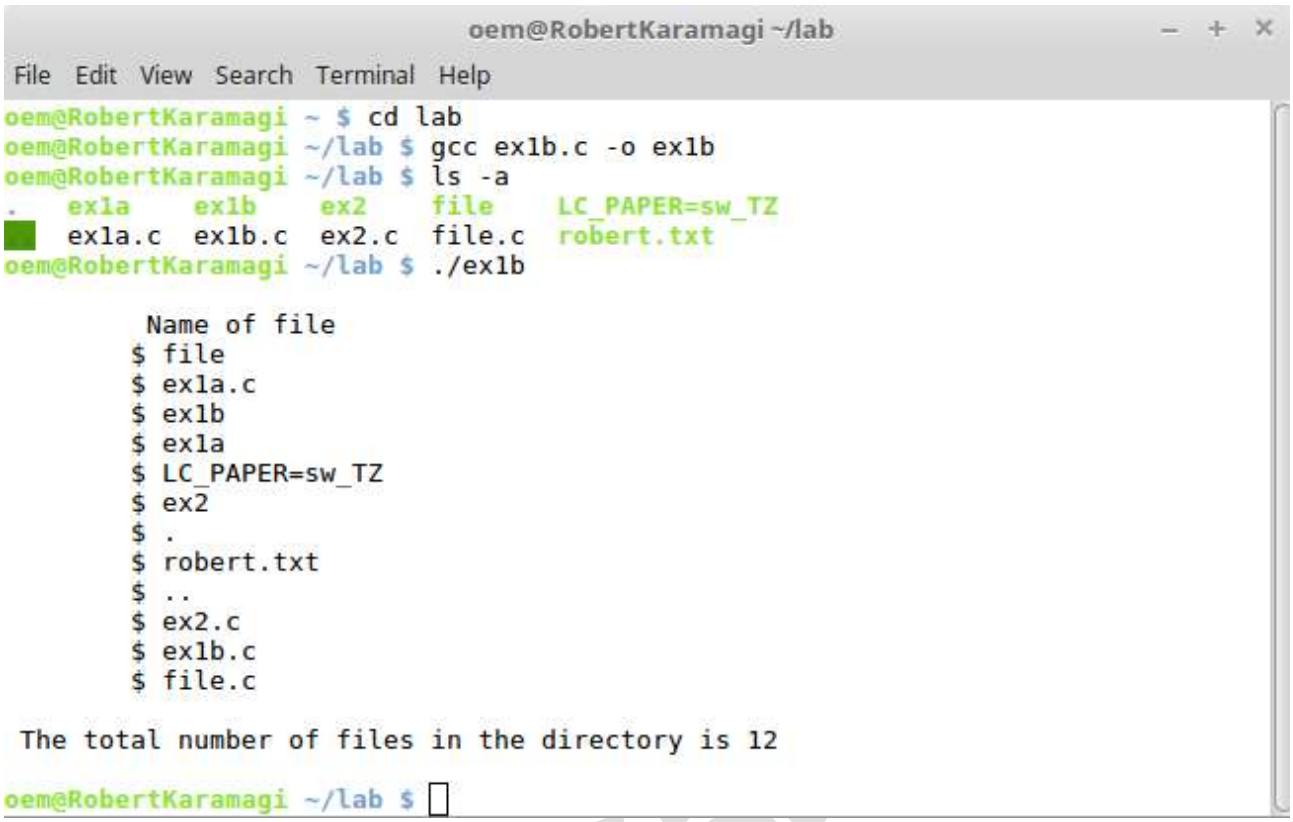
```
File  Edit  View  Search  Terminal  Help
oem@RobertKaramagi ~ $ cd lab
oem@RobertKaramagi ~/lab $ gcc ex1a.c -o ex1a
oem@RobertKaramagi ~/lab $ ./ex1a

Child process

Child process parent ID : 13981

Child process ID : 13982

Tue Nov 29 18:16:37 EAT 2016

Parent process

Parent process ID : 13981

  File: 'ex1a.c'
  Size: 615             Blocks: 8          IO Block: 4096   regular file
Device: 808h/2056d      Inode: 544511      Links: 2
Access: (0644/-rw-r--r--)  Uid: (29999/     oem)   Gid: (29999/     oem)
Access: 2016-11-29 18:16:35.985235407 +0300
Modify: 2016-11-29 18:16:26.209235157 +0300
Change: 2016-11-29 18:16:26.209235157 +0300
 Birth: -
oem@RobertKaramagi ~/lab $
```

Experiment 1b

/*UNIX System Calls: opendir,readdir (ls)*/

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<dirent.h>

int main(int argc,char* argv[])
{
        DIR *d;
        struct dirent *r;
        int i=0;
        d=opendir(".");
        if(d==NULL)
        {
                printf("Error! Unable to open directory.\n");
                exit(1);
        }
        printf("\n\t Name of file \n");
        while((r=readdir(d)) != NULL)
        {
                printf("\t$ %s \n",r->d_name);
                i=i+1;
        }
        closedir(d);
        printf("\n The total number of files in the directory is %d \n\n",i);
        return 0;
}
```

Output

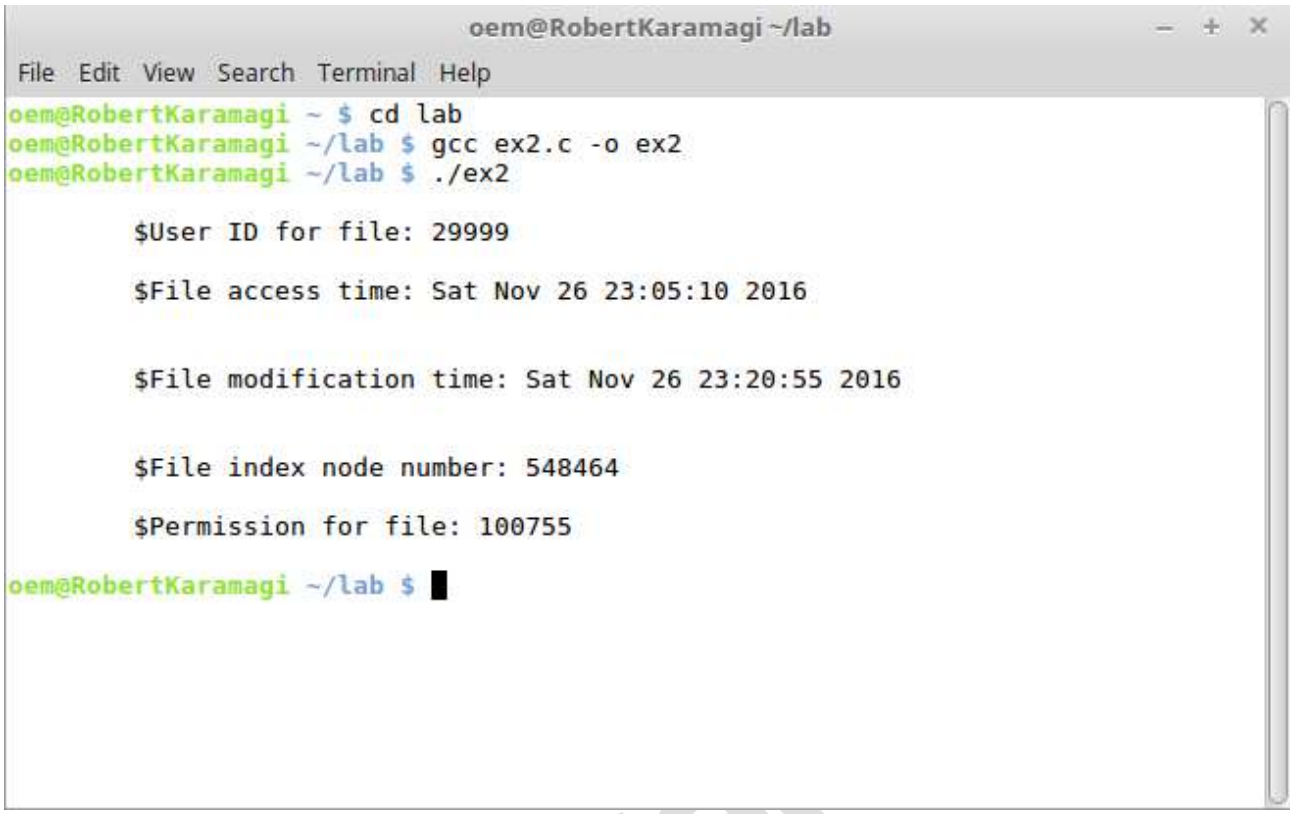


Experiment 2

/*UNIX System Calls: open, read, write*/

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

int main(int argc,char*argv[])
{
        char buf[100];
        struct stat s;
        int fd,n;
        fd=open(argv[1], O_WRONLY,0777);
        fd=creat(argv[2],O_RDONLY);
        stat(argv[2],&s);
        if(fd==-1)
        printf("Error in Creation.");
        while( ( n=read(fd,buf,sizeof(buf)) ) > 0 )
        {
                if(write(fd,buf,n)!=n)
                {
                        close(fd);

                }
        }
        printf("\n\t$User ID for file: %d \n",s.st_uid);
        printf("\n\t$File access time: %s \n",ctime(&s.st_atime));
        printf("\n\t$File modification time: %s \n",ctime(&s.st_mtime));
        printf("\n\t$File index node number: %ld \n",s.st_ino);
        printf("\n\t$Permission for file: %o \n\n",s.st_mode);
        return 0;
}
```

Output



Experiment 3

/*Unix commands: ls,grep */

```
#include <sys/types.h>
#include <sys/dir.h>
#include <sys/param.h>
#include <stdio.h>
#include <stdlib.h>
#include<string.h>

void grep();

int main(int argc, char* argv[])
{
	int n;
	while(1)
	{
		printf("\nEnter an option.\n");
		printf("\n1.Unix command ls.");
		printf("\n2.Unix command grep.");
		printf("\n3.Unix command cp.");
		printf("\n4.Unix command rm.\n");
		scanf("%d",&n);
		switch(n)
		{
			case 1:
				system("ls");
				break;

			case 2:
				grep();
				break;

			case 3:
				system("cp -l ex3.c copy.c");
				printf("The file has been copied succesfully.");
				break;

			case 4:
				system("rm copy.c");
				printf("The file has been removed succesfully.");
				break;

			default:
				printf("Invalid choice.\n");
				exit(0);
				break;

		}
	}
	return 0;

}

void grep()
{
```

```
        char fn[10],pat[10],temp[200];
        FILE *fp;
        printf("Enter file name\n");
        scanf("%s",fn);
        printf("Enter pattern to be searched\n");
        scanf("%s",pat);
        fp=fopen(fn,"r");
        while(!feof(fp))
        {
                fgets(temp,1000,fp);
                if(strstr(temp,pat))
                printf("%s",temp);
        }
        fclose(fp);
}
```

Output

```
oem@RobertKaramagi ~ $ cd lab
oem@RobertKaramagi ~/lab $ gcc ex3.c -o ex3
oem@RobertKaramagi ~/lab $ ./ex3

Enter an option.

1.Unix command ls.
2.Unix command grep.
3.Unix command cp.
4.Unix command rm.
1
date    ex1b    ex2.c  ex4a    ex4b.c  ex5b    ex6.c
ex1a    ex1b.c  ex3    ex4a.c  ex5a    ex5b.c  LC_PAPER=sw_TZ
ex1a.c  ex2     ex3.c  ex4b    ex5a.c  ex6

Enter an option.

1.Unix command ls.
2.Unix command grep.
3.Unix command cp.
4.Unix command rm.
2
Enter file name
ex3.c
Enter pattern to be searched
fp
        FILE *fp;
        fp=fopen(fn,"r");
        while(!feof(fp))
                fgets(temp,1000,fp);
        fclose(fp);

Enter an option.

1.Unix command ls.
2.Unix command grep.
3.Unix command cp.
4.Unix command rm.
3
The file has been copied succesfully.
Enter an option.
```

```
1.Unix command ls.
2.Unix command grep.
3.Unix command cp.
4.Unix command rm.
1
copy.c  ex1a.c  ex2    ex3.c   ex4b    ex5a.c  ex6
date    ex1b    ex2.c  ex4a    ex4b.c  ex5b    ex6.c
ex1a    ex1b.c  ex3    ex4a.c  ex5a    ex5b.c  LC_PAPER=sw_TZ

Enter an option.

1.Unix command ls.
2.Unix command grep.
3.Unix command cp.
4.Unix command rm.
4
The file has been removed succesfully.
Enter an option.

1.Unix command ls.
2.Unix command grep.
3.Unix command cp.
4.Unix command rm.
1
date    ex1b    ex2.c  ex4a    ex4b.c  ex5b    ex6.c
ex1a    ex1b.c  ex3    ex4a.c  ex5a    ex5b.c  LC_PAPER=sw_TZ
ex1a.c  ex2     ex3.c  ex4b    ex5a.c  ex6

Enter an option.

1.Unix command ls.
2.Unix command grep.
3.Unix command cp.
4.Unix command rm.
5
Invalid choice.
oem@RobertKaramagi ~/lab $
```

Experiment 4a

/*First Come First Served Scheduling*/

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

int main()
{
	int n, a[10], b[10], t[10], w[10], g[10], i,j,k;
	float att=0, awt=0;
	for(i=0;i<10;i++)
	a[i]=b[i]=t[i]=w[i]=g[i]=0;
	printf("\nEnter the number of processes.\n");
	scanf("%d",&n);
	printf("\nEnter the burst times.\n");
	for(i=0;i<n;i++)
	scanf("%d",&b[i]);
	printf("\nEnter the arrival times.\n");
	for(i=0;i<n;i++)
	scanf("%d",&a[i]);
	for(i=0;i<n;i++)
	g[i+1]=g[i]+b[i];
	for(i=0;i<n;i++)
	{
		w[i]=g[i]-a[i];
		t[i]=g[i+1]-a[i];
		awt=awt+w[i];
		att=att+t[i];
	}
	awt=awt/n;
	att=att/n;
	puts("\n\n+---------+------------+-------------+--------------+-----------------+");
	  puts("| Process | Burst Time | Arrival Time |  Waiting Time | Turn around Time |");
	 puts("+---------+------------+-------------+--------------+-----------------+");
	 for(i=0; i<n; i++)
	{
		printf("|   P%2d  |    %2d    |     %2d     |      %2d      |       %2d       |\n"
			,i+1,b[i],a[i],w[i],t[i]);
		puts("+---------+------------+-------------+--------------+-----------------+");
	}
	puts("\n\nGANTT CHART");
	 puts("************");

	// print top bar
	 printf(" ");
	for(i=0; i<n; i++)
	{
		for(j=0; j<b[i]; j++)
		printf("--");
		printf(" ");
	}
	printf("\n|");

	// printing process id in the middle
	for(i=0; i<n; i++)
	{
```

```
                for(j=0; j<b[i]-1; j++)
                printf(" ");
                printf("P%d", i+1);
                for(j=0; j<b[i]-1; j++)
                printf(" ");
                printf("|");
        }
         printf("\n ");

        // printing bottom bar
        for(i=0; i<n; i++)
        {
                for(j=0; j<b[i]; j++)
                printf("--");
                printf(" ");
        }
        printf("\n");

        // printing the time line
        printf("0");
        for(i=0; i<n; i++)
        {
                for(j=0; j<b[i]; j++)
                printf("  ");
                if(g[i] > 9)
                printf("\b"); // backspace : remove 1 space
                printf("%d", g[i+1]);
        }
        printf("\n");

        printf("\n\nAverage waiting time: %f ms\n",awt);
        printf("\nAverage turn around time: %f ms\n",att);
        return 0;
}
```

Output

```
oem@RobertKaramagi ~ $ cd lab
oem@RobertKaramagi ~/lab $ gcc ex4a.c -o ex4a
oem@RobertKaramagi ~/lab $ ./ex4a

Enter the number of processes.
4

Enter the burst times.
4
9
8
13

Enter the arrival times.
0
2
4
3


+---------+------------+--------------+---------------+------------------+
| Process | Burst Time | Arrival Time |  Waiting Time | Turn around Time |
+---------+------------+--------------+---------------+------------------+
|   P 1   |      4     |       0      |        0      |         4        |
+---------+------------+--------------+---------------+------------------+
|   P 2   |      9     |       2      |        2      |        11        |
+---------+------------+--------------+---------------+------------------+
|   P 3   |      8     |       4      |        9      |        17        |
+---------+------------+--------------+---------------+------------------+
|   P 4   |     13     |       3      |       18      |        31        |
+---------+------------+--------------+---------------+------------------+


GANTT CHART
***********
 -------- ------------------ ---------------- --------------------------
|   P1   |        P2        |       P3       |            P4            |
 -------- ------------------ ---------------- --------------------------
0        4                  13               21                         34


Average waiting time: 7.250000 ms

Average turn around time: 15.750000 ms
oem@RobertKaramagi ~/lab $
```

Experiment 4b

/*Shortest Job First Scheduling*/

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

int main()
{
	int i,j,n,temp,temp1,b[10],t[10],w[10],p[10];
	float att=0,awt=0;
	for(i=0;i<10;i++)
	b[i]=t[i]=w[i]=p[i]=0;
	printf("\nEnter the number of processes.\n");
	scanf("%d",&n);
	printf("\nEnter the burst times.\n");
	for(i=0;i<n;i++)
	{
		printf("P[%d] : ", i+1);
		scanf("%d", &b[i]);
	}

	for(i=0;i<n-1;i++)
	{
		for(j=i+1;j<n;j++)
		{
			if(b[i]>b[j])
			{
				temp=b[i];
				temp1=p[i];
				b[i]=b[j];
				p[i]=p[j];
				b[j]=temp;
				p[j]=temp1;
			}
		}
	}
	w[0]=0;
	for(i=0;i<n;i++)
	{
		w[i+1]=w[i]+b[i];
		t[i]=w[i]+b[i];
		awt=awt+w[i];
		att=att+t[i];
	}

	puts("\n\n+---------+------------+--------------+------------------+");
	puts("| Process | Burst Time | Waiting Time | Turn around Time |");
	puts("+---------+------------+--------------+------------------+");
	for(i=0; i<n; i++)
	{
		printf("|   P%2d  |    %2d    |      %2d      |        %2d       |\n"
			,i+1,b[i],w[i],t[i]);
		puts("+---------+------------+--------------+------------------+");
	}
	puts("\n\nGANTT CHART");
	puts("************");
```

```
        // print top bar
        printf(" ");
        for(i=0; i<n; i++)
        {
                for(j=0; j<b[i]; j++)
                printf("--");
                printf(" ");
        }
        printf("\n|");

        // printing process id in the middle
        for(i=0; i<n; i++)
        {
                for(j=0; j<b[i]-1; j++)
                printf(" ");
                printf("P%d", i+1);
                for(j=0; j<b[i]-1; j++)
                printf(" ");
                printf("|");
        }
        printf("\n ");

        // printing bottom bar
        for(i=0; i<n; i++)
        {
                for(j=0; j<b[i]; j++)
                printf("--");
                printf(" ");
        }
        printf("\n");

        // printing the time line
        printf("0");
        for(i=0; i<n; i++)
        {
                for(j=0; j<b[i]; j++)
                printf("  ");
                if(t[i] > 9)
                printf("\b"); // backspace : remove 1 space
                printf("%d", t[i]);
        }
         printf("\n");

        printf("\n\nAverage waiting time: %f ms\n",awt);
        printf("\nAverage turn around time: %f ms\n",att);
        return 0;
}
```

Output

```
File  Edit  View  Search  Terminal  Help
oem@RobertKaramagi ~ $ cd lab
oem@RobertKaramagi ~/lab $ gcc ex4b.c -o ex4b
oem@RobertKaramagi ~/lab $ ./ex4b

Enter the number of processes.
5

Enter the burst times.
P[1] : 3
P[2] : 2
P[3] : 6
P[4] : 4
P[5] : 5


+---------+------------+--------------+------------------+
| Process | Burst Time | Waiting Time | Turn around Time |
+---------+------------+--------------+------------------+
|   P 1   |      2     |       0      |          2       |
+---------+------------+--------------+------------------+
|   P 2   |      3     |       2      |          5       |
+---------+------------+--------------+------------------+
|   P 3   |      4     |       5      |          9       |
+---------+------------+--------------+------------------+
|   P 4   |      5     |       9      |         14       |
+---------+------------+--------------+------------------+
|   P 5   |      6     |      14      |         20       |
+---------+------------+--------------+------------------+


GANTT CHART
***********
 ---- ------ -------- ---------- ------------
| P1 |  P2  |   P3   |    P4    |     P5     |
 ---- ------ -------- ---------- ------------
0    2      5        9          14           20


Average waiting time: 30.000000 ms

Average turn around time: 50.000000 ms
oem@RobertKaramagi ~/lab $
```

Experiment 5a

/*Round Robin Scheduling*/

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

int main()
{
	int i,j,k,n,bt[20],gc[20],wt[20],tat[20],bt1[20],st[20],ts,count=0,sum_bt=0,tq;
	int swt=0,stat=0,temp,sq=0,c,p=0;
	float awt=0.0,atat=0.0;
	printf("\nEnter the number of processs: ");
	 scanf("%d",&n);
	printf("\nEnter the burst times:\n");
	for(i=0; i<n; i++)
	 {
		scanf("%d",&bt[i]);
		bt1[i]=bt[i];
		st[i]=bt[i];
	}

	printf("\nEnter the time slice.\n");
	scanf("%d",&ts);
	tq=ts;
	for(i=0; i<n; i++)
	sum_bt=sum_bt+bt[i];
	for(k=0; k<n; k++)
	{
		do
		{
			for(i=0; i<n; i++)
			{
				if(bt[i]>=ts)
				{
					for(j=count; j<(count+ts); j++)
					gc[j]=i+1;
					count+=ts;
					bt[i]=bt[i]-ts;
				}

				else
				{
					for(j=count; j<=(count+bt[i]); j++)
					gc[j]=i+1;
					count+=bt[i];
					bt[i]=0;
				}
			}
		}while(bt[k]!=0);
	}

	while(1)
	{
		for(i=0,count=0;i<n;i++)
		{
			temp=tq;
```

```
                if(st[i]==0)
                {
                count++;
                continue;
                }
                        if(st[i]>tq)
                        st[i]=st[i]-tq;
                        else if(st[i]>=0)
                        {
                                temp=st[i];
                                st[i]=0;
                        }
                        sq=sq+temp;
                        tat[i]=sq;
        }
        if(n==count)
        break;
}

for(i=0;i<n;i++)
{
        wt[i]=tat[i]-bt1[i];
        swt=swt+wt[i];
        stat=stat+tat[i];
}
awt=(float)swt/n;
atat=(float)stat/n;
puts("\n\n+---------+------------+--------------+-----------------+");
 puts("| Process | Burst Time | Waiting Time | Turn around Time |");
puts("+---------+------------+--------------+-----------------+");

for(i=0;i<n;i++)
{
        printf("|   %2d    |    %2d    |     %2d      |       %2d       |\n"
                ,i+1,bt1[i],wt[i],tat[i]);
        puts("+---------+------------+--------------+-----------------+");
}

printf("\nAverage waiting time: %f ms\n",awt);
printf("\nAverage Turn around time: %f ms\n",atat);
puts("\n\nGANTT CHART");
puts("***********");
c=gc[0];
printf("=========================================\n");
printf("|  P%d  |",gc[0]);
for(i=1;i<sum_bt;i++)
{
        if(gc[i]!=c)
        {
                printf("  P%d  |",gc[i]);
                c=gc[i];
        }
}
 printf("\n=========================================");
printf("\n0      ");
c=gc[0];
for(i=1;i<=sum_bt;i++)
```

```
        {
                p=p+1;
                if(gc[i]!=c)
                {
                        printf("%d      ",p);
                        c=gc[i];
                }
        }
        printf("\n");
        return 0;
}
```

Output

```
File  Edit  View  Search  Terminal  Help
oem@RobertKaramagi ~ $ cd lab
oem@RobertKaramagi ~/lab $ gcc ex5a.c -o ex5a
oem@RobertKaramagi ~/lab $ ./ex5a

Enter the number of processs: 4

Enter the burst times:
2
3
3
2

Enter the time slice.
2


+---------+------------+--------------+-------------------+
| Process | Burst Time | Waiting Time | Turn around Time |
+---------+------------+--------------+-------------------+
|    1    |      2     |       0      |         2         |
+---------+------------+--------------+-------------------+
|    2    |      3     |       6      |         9         |
+---------+------------+--------------+-------------------+
|    3    |      3     |       7      |        10         |
+---------+------------+--------------+-------------------+
|    4    |      2     |       6      |         8         |
+---------+------------+--------------+-------------------+

Average waiting time: 4.750000 ms

Average Turn around time: 7.250000 ms


GANTT CHART
***********
==========================================
|  P1  |  P2  |  P3  |  P4  |  P2  |  P3  |
==========================================
0      2      4      6      8      9      10
oem@RobertKaramagi ~/lab $
```

Experiment 5b

/*Priority Scheduling*/

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


int main()
{
	int i,j,n,temp,temp1,temp2,pr[10],b[10],t[10],w[10],p[10];
	float att=0,awt=0;
	printf("\nEnter the number of processes.");
	scanf("%d",&n);
	printf("\nEnter the burst times.\n");
	for(i=0;i<n;i++)
	{
		scanf("%d",&b[i]);
		p[i]=i;
	}
	printf("\nEnter the priority.\n");
	for(i=0;i<n;i++)
	{
		scanf("%d",&pr[i]);
	}
	for(i=0;i<n-1;i++)
	{
		for(j=i+1;j<n;j++)
		{
			if(pr[i]>pr[j])
			{
				temp=b[i];
				temp1=p[i];
				temp2=pr[i];
				b[i]=b[j];
				p[i]=p[j];
				pr[i]=pr[j];
				b[j]=temp;
				p[j]=temp1;
				pr[j]=temp2;
			}
		}
	}
	w[0]=0;
	for(i=0;i<n;i++)
	w[i+1]=w[i]+b[i];
	for(i=0;i<n;i++)
	{
		t[i]=w[i]+b[i];
		awt=awt+w[i];
		att=att+t[i];
	}
	awt=awt/n;
	att=att/n;
	puts("\n\n+---------+-----------+-------------+-----------------+");
	puts("| Process | Burst Time | Waiting Time | Turn around Time |");
	puts("+---------+-----------+-------------+-----------------+");
```

```
	for(i=0; i<n; i++)
	{
		printf("|   %2d    |    %2d    |    %2d    |       %2d       |\n"
			,i+1,b[i],w[i],t[i]);
		puts("+---------+------------+--------------+------------------+");
	}
	puts("\n\nGANTT CHART");
	puts("************");

	// print top bar
	printf(" ");
	 for(i=0; i<n; i++)
	{
		for(j=0; j<b[i]; j++)
		printf("--");
		printf(" ");
	}
	 printf("\n|");

	// printing process id in the middle
	for(i=0; i<n; i++)
	{
		for(j=0; j<b[i]-1; j++)
		printf(" ");
		printf("P%d", i+1);
		for(j=0; j<b[i]-1; j++)
		printf(" ");
		printf("|");
	}
	 printf("\n ");

	// printing bottom bar
	 for(i=0; i<n; i++)
	{
		for(j=0; j<b[i]; j++)
		printf("--");
		printf(" ");
	}
	printf("\n");

	// printing the time line
	printf("0");
	for(i=0; i<n; i++)
	{
		for(j=0; j<b[i]; j++)
		printf("  ");
		if(t[i] > 9)
		printf("\b"); // backspace : remove 1 space
		printf("%d", t[i]);

	}
	 printf("\n");

	printf("\n\nAverage waiting time: %f ms\n",awt);
	printf("\nAverage turn around time: %f ms\n",att);
	return 0;
}
```

Output

```
File  Edit  View  Search  Terminal  Help
oem@RobertKaramagi ~/lab $ gcc ex5b.c -o ex5b
oem@RobertKaramagi ~/lab $ ./ex5b

Enter the number of processes.5

Enter the burst times.
2
5
3
1
2

Enter the priority.
2
4
3
5
1


+---------+------------+--------------+-------------------+
| Process | Burst Time | Waiting Time | Turn around Time |
+---------+------------+--------------+-------------------+
|    1    |      2     |       0      |         2         |
+---------+------------+--------------+-------------------+
|    2    |      2     |       2      |         4         |
+---------+------------+--------------+-------------------+
|    3    |      3     |       4      |         7         |
+---------+------------+--------------+-------------------+
|    4    |      5     |       7      |        12         |
+---------+------------+--------------+-------------------+
|    5    |      1     |      12      |        13         |
+---------+------------+--------------+-------------------+


GANTT CHART
***********
 ---- ---- ------ ---------- --
| P1 | P2 |  P3  |    P4    |P5|
 ---- ---- ------ ---------- --
0    2    4      7         12 13


Average waiting time: 5.000000 ms

Average turn around time: 7.600000 ms
```

Experiment 6

/*Interprocess Communication*/

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

int main(int argc, char* argv[])
{
	int pid,pfd[2],n,a,b,c;
	if(pipe(pfd)==-1)
	{
		printf("\nError in pipe connection\n");
		exit(1);
	}
	pid=fork();
	if(pid>0)
	{
		puts("\nParent Process");
		puts("\n***************");
		printf("\n\n\tFibonacci Series");
		printf("\nEnter the limit for the series:");
		scanf("%d",&n);
		write(pfd[1],&n,sizeof(n));
		close(pfd[1]);
		exit(0);
	}
	else
	{
		read(pfd[0],&n,sizeof(n));
		puts("\nChild Process");
		puts("\n***************");
		a=0;
		b=1;
		close(pfd[0]);
		printf("\nFibonacci Series is:");
		printf("\n\n%d\n%d",a,b);
		while(n>2)
		{
			c=a+b;
			printf("\n%d",c);
			a=b;
			b=c;
			n--;
		}

	}
	puts("\n");
	return 0;

}
```

Output

```
File  Edit  View  Search  Terminal  Help
oem@RobertKaramagi ~ $ cd lab
oem@RobertKaramagi ~/lab $ gcc ex6.c -o ex6
oem@RobertKaramagi ~/lab $ ./ex6

Parent Process

**************


        Fibonacci Series
Enter the limit for the series:25

Child Process

**************

Fibonacci Series is:

0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
10946
17711
28657
46368
```

Experiment 7

/*Producer-Consumer Problem using Semaphores*/

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

int mutex=1,full=0,empty=3,x=0;
int main()
{
        int n;
        void producer();
        void consumer();
        int wait(int);
        int signal(int);
        printf("\n 1.Producer  \n 2.Consumer \n 3.Exit");
        while(1)
        {
                printf("\n\n Enter your choice:");
                scanf("%d",&n);
                switch(n)
                {
                        case 1:
                                if((mutex==1)&&(empty!=0))
                                producer();
                                else
                                printf(" Buffer is full");
                                break;

                        case 2:
                                if((mutex==1)&&(full!=0))
                                consumer();
                                else
                                printf(" Buffer is empty");
                                break;

                        case 3:
                                exit(0);
                                break;
                }
        }
}

int wait(int s)
{
        return (--s);
}

int signal(int s)
{
        return(++s);
}

void producer()
{
```

```
        mutex=wait(mutex);
        full=signal(full);
        empty=wait(empty);
        x++;
        printf(" Producer produces the item %d",x);
        mutex=signal(mutex);
}

void consumer()
{
        mutex=wait(mutex);
        full=wait(full);
        empty=signal(empty);
        printf(" Consumer consumes item %d",x);
        x--;
        mutex=signal(mutex);
}
```

Output

```
File  Edit  View  Search  Terminal  Help
oem@RobertKaramagi ~ $ cd lab
oem@RobertKaramagi ~/lab $ gcc ex7.c -o ex7
oem@RobertKaramagi ~/lab $ ./ex7

 1.Producer
 2.Consumer
 3.Exit

 Enter your choice:1
 Producer produces the item 1

 Enter your choice:1
 Producer produces the item 2

 Enter your choice:1
 Producer produces the item 3

 Enter your choice:1
 Buffer is full

 Enter your choice:2
 Consumer consumes item 3

 Enter your choice:2
 Consumer consumes item 2

 Enter your choice:2
 Consumer consumes item 1

 Enter your choice:2
 Buffer is empty

 Enter your choice:3
oem@RobertKaramagi ~/lab $
```

Experiment 8

/*First Fit Memory Management Scheme*/

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

struct firstfit
{

        int b[20],p[20];

} *ptr,*ptr1;

int i,j,n,m;
void display(struct firstfit *disp);

int main()
{

        printf("\nEnter the number of Blocks: ");
        scanf("%d",&n);
        ptr=(struct firstfit*)malloc(n*sizeof(struct firstfit));
        printf("\nEnter the size of the Blocks.\n");
        for(i=0;i<n;i++)
        {
                printf("Block [%d]: ",i);
                scanf("%d", &(ptr+i)->b[i]);
        }
        printf("\nEnter the number of Processes: ");
        scanf("%d",&m);
        ptr1=(struct firstfit*)malloc(m*sizeof(struct firstfit));
        printf("\nEnter the size of the Processes.\n");
        for(i=0;i<m;i++)
        {
                printf("Process [%d]: ",i);
                scanf("%d", &(ptr1+i)->p[i]);
        }

        struct firstfit disp;
        display(&disp);

}

void display(struct firstfit *disp)
{
        for(i=0;i<n;i++)
        {
                for(j=0;j<m;j++)
                {
                        if((ptr1+j)->p[j] <= (ptr+i)->b[i])
                        {
                                printf("\nThe Process %d is allocated to Block %d\n",j, i);
                                (ptr1+j)->p[j]=10000;
                                break;
                        }
                }
```

```
        }

        for(j=0;j<m;j++)
        {
                if((ptr1+j)->p[j] != 10000)
                {
                        printf("\nThe Process %d is not allocated\n",j);
                }
        }
}
```

Output

```
File  Edit  View  Search  Terminal  Help
oem@RobertKaramagi ~ $ cd lab
oem@RobertKaramagi ~/lab $ gcc ex8.c -o ex8
oem@RobertKaramagi ~/lab $ ./ex8

Enter the number of Blocks: 5

Enter the size of the Blocks.
Block [0]: 10
Block [1]: 20
Block [2]: 10
Block [3]: 15
Block [4]: 25

Enter the number of Processes: 5

Enter the size of the Processes.
Process [0]: 20
Process [1]: 10
Process [2]: 15
Process [3]: 10
Process [4]: 35

The Process 1 is allocated to Block 0

The Process 0 is allocated to Block 1

The Process 3 is allocated to Block 2

The Process 2 is allocated to Block 3

The Process 4 is not allocated
oem@RobertKaramagi ~/lab $
```

Experiment 9a

/*Best Fit Memory Management Scheme*/

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

struct bestfit
{

        int b[20],p[20],b1[20],p1[20];

}*ptr,*ptr1;

int i,j,n,m;
void display(struct bestfit *disp);

int main()
{
        int temp,temp1,temp2;
        printf("\nEnter the number of Blocks: ");
        scanf("%d",&n);
        ptr=(struct bestfit*)malloc(n*sizeof(struct bestfit));
        printf("\nEnter the size of the Blocks.\n");
        for(i=0;i<n;i++)
        {
                printf("Block [%d]: ",i);
                scanf("%d", &(ptr+i)->b[i]);
                (ptr+i)->b1[i]=i;
        }
        printf("\nEnter the number of Processes: ");
        scanf("%d",&m);
        ptr1=(struct bestfit*)malloc(m*sizeof(struct bestfit));
        printf("\nEnter the size of the Processes.\n");
        for(i=0;i<m;i++)
        {
                printf("Process [%d]: ",i);
                scanf("%d", &(ptr1+i)->p[i]);
                (ptr1+i)->p1[i]=i;
        }


        struct bestfit disp;
        display(&disp);
}

void display(struct bestfit *disp)
{

        for(i=0;i<n;i++)
        {
                for(j=0;j<m;j++)
                {

                        if((ptr1+j)->p[j] <= (ptr+i)->b[i])
                        {
```

```
                                printf("\nThe Process %d is allocated to
                                Block %d\n",(ptr1+j)->p1[j],(ptr+i)->b1[i]);
                                (ptr1+j)->p[j]=10000;
                                break;
                        }
                }
        }
        for(j=0;j<m;j++)
        {
                if((ptr1+j)->p[j] != 10000)
                {
                        printf("\nThe Process %d is not allocated\n",j);
                }
        }
}
```

Output

```
File  Edit  View  Search  Terminal  Help
oem@RobertKaramagi ~ $ cd lab
oem@RobertKaramagi ~/lab $ gcc ex9a.c -o ex9a
oem@RobertKaramagi ~/lab $ ./ex9a

Enter the number of Blocks: 5

Enter the size of the Blocks.
Block [0]: 10
Block [1]: 20
Block [2]: 30
Block [3]: 40
Block [4]: 50

Enter the number of Processes: 5

Enter the size of the Processes.
Process [0]: 47
Process [1]: 28
Process [2]: 36
Process [3]: 4
Process [4]: 15

The Process 3 is allocated to Block 0

The Process 4 is allocated to Block 1

The Process 1 is allocated to Block 2

The Process 2 is allocated to Block 3

The Process 0 is allocated to Block 4
oem@RobertKaramagi ~/lab $
```

Experiment 9b

/*Worst Fit Memory Management Scheme*/

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

struct worstfit
{

        int p[20],p1[20],p2[20],b[20],frag[20],bf[20],ff[20];

} *ptr, wf;

int i,m;
void display(struct worstfit *disp);

int main()
{
        int j,n,temp,highest=0;
        printf("\nEnter the number of Blocks: ");
        scanf("%d",&n);
        printf("\nEnter the size of the Blocks.\n");
        for(i=1;i<=n;i++)
        {
                printf("Block [%d]: ",i);
                scanf("%d", &wf.b[i]);
        }
        printf("\nEnter the number of Processes: ");
        scanf("%d",&m);
        ptr=(struct worstfit*)malloc(m*sizeof(struct worstfit));
        printf("\nEnter the size of the Processes.\n");
        for(i=1;i<=m;i++)
        {
                printf("Process [%d]: ",i);
                scanf("%d", &wf.p[i]);
                (ptr+i)->p1[i]=wf.p[i];
                (ptr+i)->p2[i]=i;
        }
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=m;j++)
                {
                        if(wf.bf[j]!=1)
                        {
                                temp= wf.b[j]-wf.p[i];
                                if(temp>=0)
                                if(highest<temp)
                                {
                                        wf.ff[i]=j;
                                        highest=temp;
                                }
                        }
                }
                wf.frag[i]=highest;
                wf.bf[wf.ff[i]]=1;
                highest=0;
```

```
        }
        struct worstfit disp;
        display(&disp);
        return 0;
}

void display(struct worstfit *disp)
{
        puts("\n\n+-------+------------+-----+----------+--------+");
        puts("|Process|Process size|Block|Block Size|Fragment|");
        puts("+-------+------------+-----+----------+--------+");
        for(i=1;i<=m;i++)
        {
                printf("| %2d  |    %2d    | %2d |   %2d   |  %2d  |"
                                ,(ptr+i)->p2[i],(ptr+i)->p1[i],wf.ff[i], wf.b[wf.ff[i]]
                                ,wf.frag[i]);
                puts("\n+-------+------------+-----+----------+--------+");
        }
        printf("\n");
}
```

Output

```
File  Edit  View  Search  Terminal  Help
oem@RobertKaramagi ~ $ cd lab
oem@RobertKaramagi ~/lab $ gcc ex9b.c -o ex9b
oem@RobertKaramagi ~/lab $ ./ex9b

Enter the number of Blocks: 3

Enter the size of the Blocks.
Block [1]: 20
Block [2]: 30
Block [3]: 10

Enter the number of Processes: 3

Enter the size of the Processes.
Process [1]: 10
Process [2]: 14
Process [3]: 6


+-------+------------+-----+----------+--------+
|Process|Process size|Block|Block Size|Fragment|
+-------+------------+-----+----------+--------+
|   1   |     10     |  2  |    30    |   20   |
+-------+------------+-----+----------+--------+
|   2   |     14     |  1  |    20    |    6   |
+-------+------------+-----+----------+--------+
|   3   |      6     |  3  |    10    |    4   |
+-------+------------+-----+----------+--------+

oem@RobertKaramagi ~/lab $
```

Experiment 10

/*Contiguous File Allocation*/

```
#include<stdio.h>
#include<stdlib.h>
int main()
{

	int n, fc[20], mb[100], i, j, k, fb[100], fs[20], mc=0;
	printf("\nEnter the number of files: ");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("\nEnter the capacity of file %d: ",i+1);
		scanf("%d",&fc[i]);
		printf("\nEnter the starting address of file %d: ",i+1);
		scanf("%d",&fs[i]);
	}
	printf("\n\nCONTIGUOUS FILE ALLOCATION");
	printf("\n**************************\n");
	for(i=0;i<100;i++)
	fb[i]=1;
	for(i=0;i<n;i++)
	{
		j=fs[i];
		{
			if(fb[j]==1)
			{
				for(k=j;k<(j+fc[i]);k++)
				{
					if(fb[k]==1)
					mc++;
				}
				if(mc==fc[i])
				{
					for(k=fs[i];k<(fs[i]+fc[i]);k++)
					{
						fb[k]=0;
					}
					printf("\nFile %d is allocated from address %d t %d.\n"
						,i+1,fs[i],fs[i]+fc[i]-1);
				}
			}
			else
			printf("\nFile %d is not allocated since contiguous memory of size %d is not
				available from address %d.\n",i+1,fc[i],fs[i]);
		}
		mc=0;
	}
	printf("\n\n");
	return 0;
}
```

Output

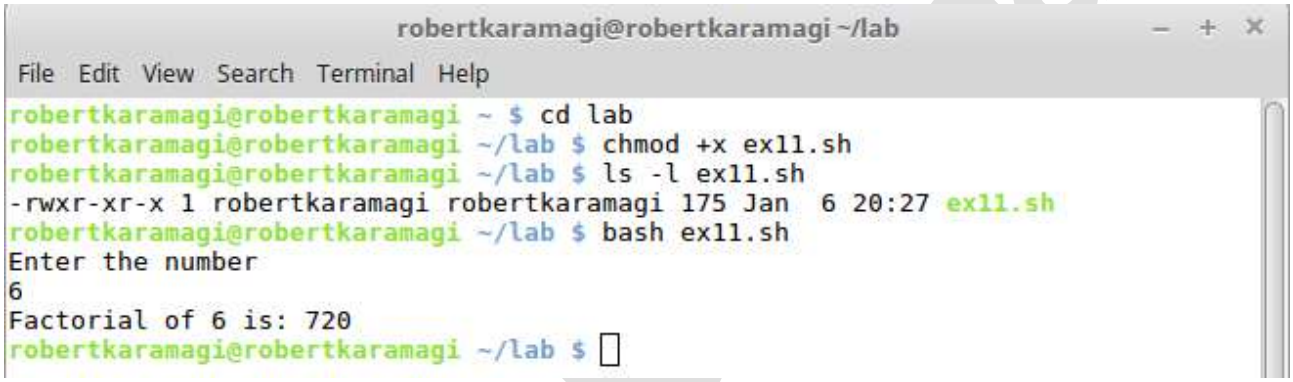
```
File  Edit  View  Search  Terminal  Help
oem@RobertKaramagi ~ $ cd lab
oem@RobertKaramagi ~/lab $ gcc ex10.c -o ex10
oem@RobertKaramagi ~/lab $ ./ex10

Enter the number of files: 4

Enter the capacity of file 1: 10

Enter the starting address of file 1: 0

Enter the capacity of file 2: 20

Enter the starting address of file 2: 10

Enter the capacity of file 3: 20

Enter the starting address of file 3: 25

Enter the capacity of file 4: 10

Enter the starting address of file 4: 40


CONTIGUOUS FILE ALLOCATION
**************************

File 1 is allocated from address 0 to 9.

File 2 is allocated from address 10 to 29.

File 3 is not allocated since contiguous memory of size 20 is not available from address 25.

File 4 is allocated from address 40 to 49.
```

Experiment 11

#Factorial of a given number

```
#!bin/bash
echo "Enter the number"
read n
i=1
fact=1
while [ $i -le $n ]
do
((fact=$fact*$i))
((i=$i+1))
done
echo "Factorial of $n is: $fact"
```

Output

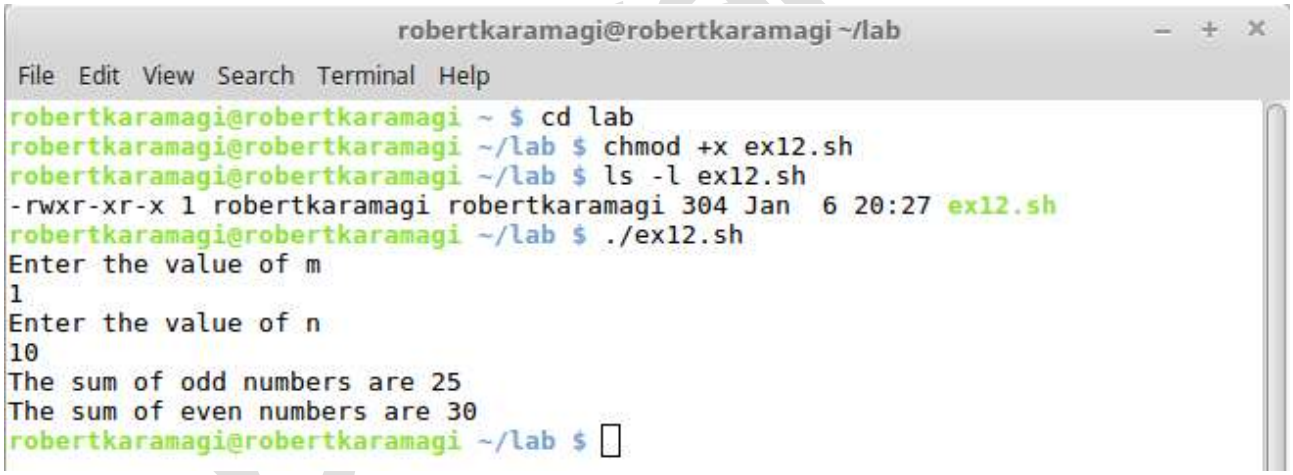


Experiment 12

Sum of Odd and Even numbers from M to N

```
#!bin/bash
echo "Enter the value of m"
read m
echo "Enter the value of n"
read n
k=0
p=0
while [ $m -le $n ]
do
((t=m%2))
if [ $t -eq 0 ]
then ((p=p+m))
else
((k=k+m))
fi
((m=m+1))
done
echo "The sum of odd numbers are $k"
echo "The sum of even numbers are $p"
```

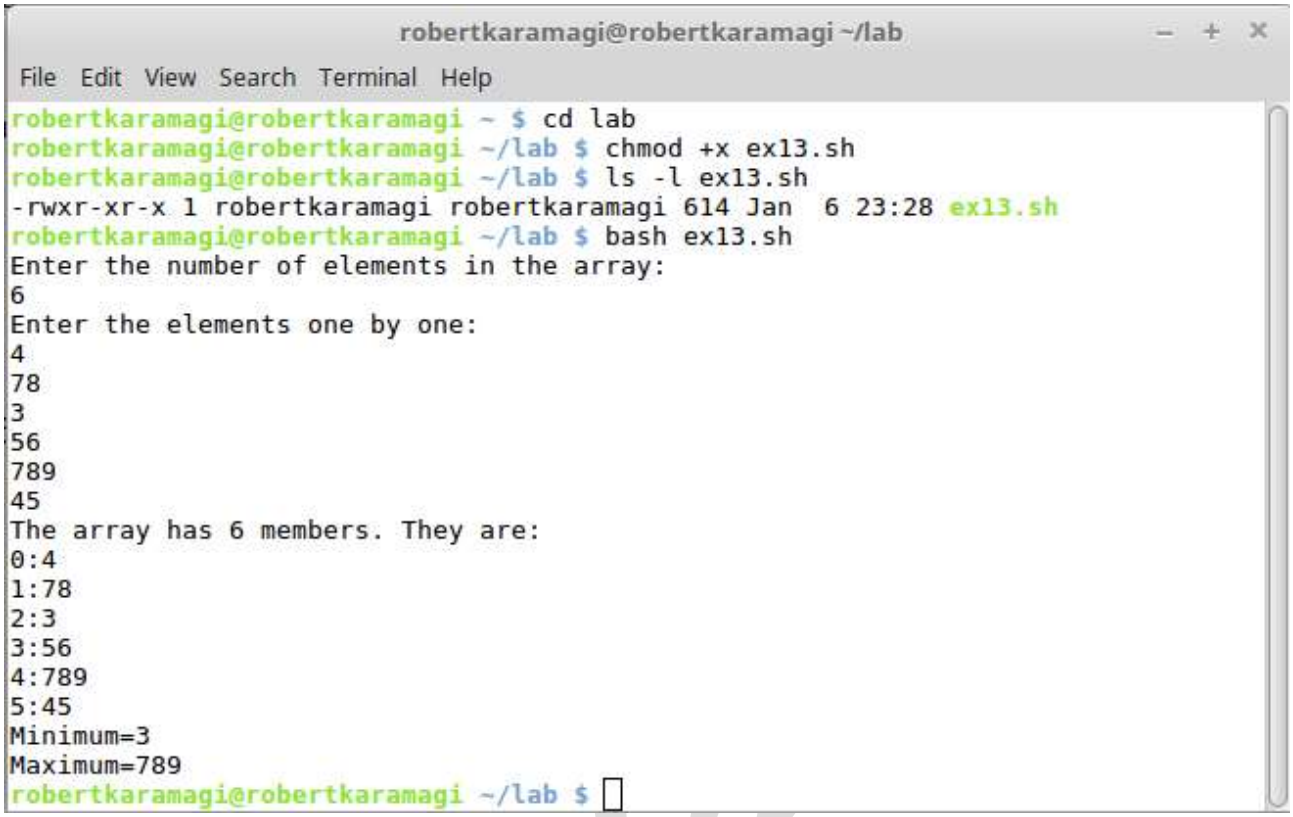
Output



Experiment 13

```
#Find out the max and min number of a given series
#!bin/bash

echo "Enter the number of elements in the array:"
read n
echo "Enter the elements one by one:"
i=0
while [ $i -lt $n ]
do
read array[$i]
let i++
done
len=${#array[*]}
echo "The array has $len members. They are: "
i=0
while [ $i -lt $len ]
do
echo "$i:${array[$i]}"
let i++
done
i=0
j=1
min=${array[$i]}
max=${array[$i]}
while [ $j -lt $len ]
do
sec=${array[$j]}
if [ $min -gt $sec ]
then
min=$sec
elif [ $max -lt $sec ]
then
max=$sec
fi
let j++
done
echo "Minimum=$min"
echo "Maximum=$max"
```

Output



Experiment 14

```
# Implementation of Calculator application
#!bin/bash

j=1
while [ $j -eq 1 ]
do
echo "Enter the first operand:"
read f1
echo "Enter the second operand:"
read f2
echo "1-> Addition"
echo "2-> Subtraction"
echo "3-> Multiplication"
echo "4-> Division"
echo "Enter your choice"
read n
case "$n" in
1)
echo "Addition"
f3=$((f1+f2))
echo "The result is:$f3"
;;
2)
echo "Subtraction"
let "f4=$f1-$f2"
echo "The result is:$f4"
;;
3)
echo "Multiplication"
let "f5=$f1 * $f2"
echo "The result is:$f5"
;;
4)
echo "Division"
let "f6=$f1 / $f2"
echo "The result is:$f6"
;;
esac
echo "Do you want to continue(press:1 otherwise press any other integer to quit)"
read j
done
```

Output

```
File  Edit  View  Search  Terminal  Help
robertkaramagi@robertkaramagi ~ $ cd lab
robertkaramagi@robertkaramagi ~/lab $ chmod +x ex14.sh
robertkaramagi@robertkaramagi ~/lab $ bash ex14.sh
Enter the first operand:
23
Enter the second operand:
23
1-> Addition
2-> Subtraction
3-> Multiplication
4-> Division
Enter your choice
1
Addition
The result is:46
Do you want to continue(press:1 otherwise press any other integer to quit)
1
Enter the first operand:
20
Enter the second operand:
2
1-> Addition
2-> Subtraction
3-> Multiplication
4-> Division
Enter your choice
2
Subtraction
The result is:18
Do you want to continue(press:1 otherwise press any other integer to quit)
1
Enter the first operand:
24
Enter the second operand:
2
1-> Addition
2-> Subtraction
3-> Multiplication
4-> Division
Enter your choice
3
Multiplication
The result is:48
Do you want to continue(press:1 otherwise press any other integer to quit)
1
Enter the first operand:
24
Enter the second operand:
12
1-> Addition
2-> Subtraction
3-> Multiplication
4-> Division
Enter your choice
4
Division
The result is:2
Do you want to continue(press:1 otherwise press any other integer to quit)
0
robertkaramagi@robertkaramagi ~/lab $
```