✅ **SET-1**

A) Shell Script – Mark Sheet

```
#!/bin/sh
echo "Enter marks of 3 subjects:"
read m1
read m2
read m3

total=$((m1+m2+m3))
avg=$((total/3))

echo "Total = $total"
echo "Average = $avg"

if [ $avg -ge 90 ]; then
 echo "Grade: O"
elif [ $avg -ge 80 ]; then
 echo "Grade: E"
elif [ $avg -ge 70 ]; then
 echo "Grade: A"
elif [ $avg -ge 60 ]; then
 echo "Grade: B"
elif [ $avg -ge 50 ]; then
 echo "Grade: C"
elif [ $avg -ge 40 ]; then
 echo "Grade: D"
else
 echo "Grade: F"
fi
```

B) Factorial

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

✅ **SET-2**

A) Prime Number

```
#!/bin/sh
echo "Enter number:"
read n
flag=0
for i in `seq 2 $((n/2))`
do
 if [ $((n%i)) -eq 0 ]; then
  flag=1
 fi
done
if [ $flag -eq 0 ]; then
 echo "Prime"
else
 echo "Not Prime"
fi
```

B) Zombie & Orphan Process (C)

```
#include<stdio.h>
#include<unistd.h>
int main(){
 if(fork()==0){
  sleep(5);
 }
 else{
  sleep(10);
 }
 return 0;
}
```

✅ **SET-3**

A) Show .sh files

```
#!/bin/sh
ls *.sh
```

B) Greatest of 3

```
#!/bin/sh
echo "Enter 3 numbers:"
read a b c
if [ $a -gt $b ] && [ $a -gt $c ]; then
 echo "$a is greatest"
elif [ $b -gt $c ]; then
 echo "$b is greatest"
else
 echo "$c is greatest"
fi
```

✅ **SET-4**

A) FCFS Scheduling (C)

```
#include<stdio.h>
int main(){
 int n,i;
 int bt[10],wt[10],tat[10];
 wt[0]=0;
 printf("Enter number of processes:");
 scanf("%d",&n);
 for(i=0;i<n;i++)
  scanf("%d",&bt[i]);

 for(i=1;i<n;i++)
  wt[i]=wt[i-1]+bt[i-1];

 for(i=0;i<n;i++){
  tat[i]=wt[i]+bt[i];
  printf("WT=%d TAT=%d\n",wt[i],tat[i]);
 }
}
```

B) Hello World

```
#!/bin/sh
echo "Hello World"
```

✅ **SET-5**

Parent-Child with kill

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

int main(){
 pid_t pid=fork();
```

```
 if(pid==0){
  while(1){
   printf("Welcome\n");
   usleep(50000);
  }
 }
 else{
  sleep(1);
  kill(pid,SIGKILL);
 }
}
```

✅ **SET-6**

A) Commands

- pwd → shows current directory
- ls → list files
- date → show date
- echo → print message

B) UID, PID, PPID

```
#include<stdio.h>
#include<unistd.h>
int main(){
 printf("UID=%d\n",getuid());
 printf("PID=%d\n",getpid());
 printf("PPID=%d\n",getppid());
}
```

✅ **SET-7**

A) Calculator

```
#!/bin/sh
echo "1.Add 2.Sub 3.Mul 4.Div"
read ch
echo "Enter two numbers:"
read a b
case $ch in
1) echo $((a+b));;
2) echo $((a-b));;
3) echo $((a*b));;
4) echo $((a/b));;
esac
```

B) Fibonacci

```
#!/bin/sh
a=0
b=1
for i in `seq 1 10`
do
 echo $a
 c=$((a+b))
 a=$b
 b=$c
done
```

✅ **SET-8**

SIGSTOP, SIGCONT, SIGKILL

```
#include<stdio.h>
#include<unistd.h>
#include<signal.h>
int main(){
 pid_t pid=fork();
 if(pid==0){
```

```
  while(1) printf("Running\n");
 }
 else{
  sleep(2);
  kill(pid,SIGSTOP);
  sleep(2);
  kill(pid,SIGCONT);
  sleep(2);
  kill(pid,SIGKILL);
 }
}
```

✅ **SET-9**

A) Sort numbers

```
#!/bin/sh
echo "Enter numbers:"
read arr
echo $arr | tr " " "\n" | sort -n
```

B) Sum

```
#!/bin/sh
sum=0
for i in 1 2 3 4 5
do
 sum=$((sum+i))
done
echo "Sum=$sum"
```

✅ **SET-10**

SJF Scheduling

```
#include<stdio.h>
int main(){
 int n,bt[10],i,j,temp;
 scanf("%d",&n);
 for(i=0;i<n;i++) scanf("%d",&bt[i]);
 for(i=0;i<n;i++)
  for(j=i+1;j<n;j++)
   if(bt[i]>bt[j]){
    temp=bt[i]; bt[i]=bt[j]; bt[j]=temp;
   }
 printf("SJF Order:");
 for(i=0;i<n;i++) printf("%d ",bt[i]);
}
```

✅ **SET-11**

A) Prime 1–100

```
#!/bin/sh
for n in `seq 2 100`
do
 f=0
 for i in `seq 2 $((n/2))`
 do
  if [ $((n%i)) -eq 0 ]; then f=1; fi
 done
 if [ $f -eq 0 ]; then echo $n; fi
done
```

B) execl()

```
#include<unistd.h>
int main(){
 execl("/bin/ls","ls",NULL);
}
```

✅ **SET-12**

Pipe

```
#include<stdio.h>
#include<unistd.h>
int main(){
 int fd[2];
 char msg[20];
 pipe(fd);
 if(fork()==0){
  read(fd[0],msg,20);
  printf("%s",msg);
 }
 else{
  write(fd[1],"Hello Pipe",10);
 }
}
```

✅ **SET-13 / 14 / 15 / 16 / 17 / 18 / 19**

👉 **Repeat programs already given above**:

- Zombie/Orphan → same C code
- UID/PID → same code
- Count digits / Sum of digits → simple shell loop
- Palindrome / Armstrong → standard shell scripts
- Sorting & Sum → same as Set-9