OS 10 marks question

Slip 1st

Write a program that demonstrates the use of nice() system call. After a child process is started using fork(), assign higher priority to the child using nice() system call

```
#include <stdio.h>
#include <unistd.h>
main()
{
   int pid;
   int retnice;
   pid = fork();
   if (pid == 0)
   {
      retnice = nice(-1);
      printf("Child process id is %d\n", getpid());
      printf("Priority value is %d\n", retnice);
  }
  else
   {  retnice = nice(15);
      printf("Child process id is %d\n", getpid());
      printf("Priority value is %d\n", retnice);
   }
   return 0;
}
```

Slip 2

Q.1 Create a child process using fork(), display parent and child process id. Child process will display the message “Hello World” and the parent process should display “Hi”.

[10 marks]

```
#include<stdio.h>
#include<unistd.h>
main()
{
 int pid;
 pid=fork();
 if(pid==0)
 {
  printf("This is Child Process\n");
  printf("Hello World\n");
  printf("Child process id is %d\n",getpid());
 }
 else
 {
  printf("This is Parent Process\n");
  printf("Hii\n");
  printf("Parent process id is %d\n",getpid());
 }
 return 0;
}
```

Slip 3

Q. 1 Creating a child process using the command exec(). Note down process ids of the parent and the child processes, check whether the control is given back to the parent after the child process terminates. [10 marks]

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
int main(int ac, char *av[])
{
   int pid;
   pid = fork();
   if (pid == 0)
   {
      printf("Child Complete\n");
      printf("Child process id is %d\n ", getpid());
      execv("/bin/ls", av);
   }
   else
   {
      printf("This is Parent Process\n");
      printf("Parent process id is %d\n", getpid());
   }
   return 0;
}
```

Slip 4

Q.1 Write a program to illustrate the concept of orphan process (Using fork() and sleep())

[10 marks]

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
int main()
{
 int pid = fork();
 if (pid > 0)
 {
  printf("Parent process\n");
  printf("ID:%d\n\n", getpid());
 }
 else if (pid == 0)
 {
  printf("Child process\n");
  printf("ID:%d\n\n", getpid());
  printf("Parent-ID:%d\n\n", getppid());
  sleep(10);

  printf("Child process\n");
  printf("ID:%d\n\n", getpid());
  printf("Parent-ID:%d\n\n", getppid());
 }
 else
 {
  printf("Failed to create child process");
 }
 return 0;
}
```

Slip 5

Q.1 Write a program that demonstrates the use of nice () system call. After a child process is started using fork (), assign higher priority to the child using nice () system call.

[10 marks]

```
#include <stdio.h>
#include <unistd.h>
main()
{
   int pid;
   int retnice;
   pid = fork();
   if (pid == 0)
   {
      retnice = nice(-1);
      printf("Child process id is %d\n", getpid());
      printf("Priority value is %d\n", retnice);
   }
   else
   {
      retnice = nice(15);
      printf("Child process id is %d\n", getpid());
      printf("Priority value is %d\n", retnice);
   }
   return 0;
}
```

Slip 6

Q.1 Write a program to find the execution time taken for execution of a given set of instructions (use clock() function)

[10 marks]

Slip 7

Q.1 Write a program to create a child process using fork().The parent should goto sleep state and child process should begin its execution. In the child process, use execl() to execute the “ls”

command

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
main()
{
   int pid;
   pid = fork();
   if (pid == 0)
   {
      printf("Child process is running\n ");
      execl("/bin/ls", "ls", NULL);
   }
   else
   {
      printf("Child Process is terminated\n");
      sleep(2);
   }
   return 0;
}
```

Slip 8

Q.1 Write a C program to accept the number of process and resources and find the need matrix content and display it.

```
include<stdio.h>
int main()
{
 int Max[10][10],need[10][10],alloc[10][10],avail[10][10];
 int p,r,i,j;

 printf("Enter the no.of.processes:");
 scanf("%d",&p);

 printf("Enter the no.of.resources:");
 scanf("%d",&r);

 printf("Enter the Max Matrix:");
 for(i=0;i<p;i++)
 {
    printf("\nFor Process %d:",i+1);
   for(j=0;j<r;j++)
    scanf("%d",&Max[i][j]);
 }
```

```
 printf("Enter the Allocation Matrix:");
 for(i=0;i<p;i++)
 {
  printf("\nFor Process %d:",i+1);
   for(j=0;j<r;j++)
    scanf("%d",&alloc[i][j]);
 }


 printf("Need Matrix :\n");
 for(i=0;i<p;i++)
 {
   for(j=0;j<r;j++){
   need[i][j]=Max[i][j] - alloc[i][j];
  printf("%d\t",need[i][j]);
 }
 printf("\n");
 }
 return 0;
}
```

Slip 9

Q.1 Write a program to create a child process using fork().The parent should goto sleep state and child process should begin its execution. In the child process, use execl() to execute the “ls” command.

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
main()
{
   int pid;
   pid = fork();
   if (pid == 0)
   {
      printf("Child process is running\n ");
      execl("/bin/ls", "ls", NULL);
   }
   else
   {
      printf("Child Process is terminated\n");
      sleep(2);
   }
   return 0;
```

```
}
```

Slip 10

Q.1 Write a program to illustrate the concept of orphan process (Using fork() and sleep())

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
int main()
{
 int pid = fork();
 if (pid > 0)
 {
  printf("Parent process\n");
  printf("ID:%d\n\n", getpid());
 }
 else if (pid == 0)
 {
  printf("Child process\n");
  printf("ID:%d\n\n", getpid());
  printf("Parent-ID:%d\n\n", getppid());
  sleep(10);

  printf("Child process\n");
  printf("ID:%d\n\n", getpid());
  printf("Parent-ID:%d\n\n", getppid());
 }
 else
 {
  printf("Failed to create child process");
 }
 return 0;
}
```

Slip 11

Q.1 Create a child process using fork(), display parent and child process id. Child process will

display the message “Hello World” and the parent process should display “Hi”

```
#include<stdio.h>
#include<unistd.h>
main()
{
 int pid;
 pid=fork();
 if(pid==0)
 {
  printf("This is Child Process\n");
  printf("Hello World\n");
  printf("Child process id is %d\n",getpid());
 }
 else
 {
  printf("This is Parent Process\n");
  printf("Hii\n");
  printf("Parent process id is %d\n",getpid());
 }
 return 0;
}
```

Slip 12

Q.1 [10] Write a program to illustrate the concept of orphan process (Using fork() and sleep()

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
int main()
{
 int pid = fork();
 if (pid > 0)
 {
  printf("Parent process\n");
  printf("ID:%d\n\n", getpid());
 }
 else if (pid == 0)
 {
  printf("Child process\n");
  printf("ID:%d\n\n", getpid());
  printf("Parent-ID:%d\n\n", getppid());
  sleep(10);

  printf("Child process\n");
  printf("ID:%d\n\n", getpid());
  printf("Parent-ID:%d\n\n", getppid());
 }
 else
 {
  printf("Failed to create child process");
 }
 return 0;
}
```

Slip 13

Q.1 Write a program that demonstrates the use of nice() system call. After a child process is started using fork(), assign higher priority to the child using nice() system call.

```
#include <stdio.h>
#include <unistd.h>
main()
{
   int pid;
   int retnice;
   pid = fork();
   if (pid == 0)
   {
      retnice = nice(-1);
      printf("Child process id is %d\n", getpid());
      printf("Priority value is %d\n", retnice);
  }
  else
   {  retnice = nice(15);
      printf("Child process id is %d\n", getpid());
      printf("Priority value is %d\n", retnice);
   }
   return 0;
}
```

Slip 14

Q.1 Write a program to find the execution time taken for execution of a given set of instructions (use clock() function)

Slip 15

Q.1 Write a program to create a child process using fork().The parent should goto sleep state and child process should begin its execution. In the child process, use execl() to execute the “ls” command

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
main()
{
   int pid;
   pid = fork();
   if (pid == 0)
   {
      printf("Child process is running\n ");
      execl("/bin/ls", "ls", NULL);
   }
   else
   {
      printf("Child Process is terminated\n");
      sleep(2);
   }
   return 0;
}
```

Slip 16

Q.1 Write a program to find the execution time taken for execution of a given set of instructions (use clock() function)

Slip 17

Q.1 Write the program to calculate minimum number of resources needed to avoid deadlock.

Slip 18

Q. 1 Write a C program to accept the number of process and resources and find the need matrix content and display it.

```
include<stdio.h>
int main()
{
 int Max[10][10],need[10][10],alloc[10][10],avail[10][10];
 int p,r,i,j;

 printf("Enter the no.of.processes:");
 scanf("%d",&p);

 printf("Enter the no.of.resources:");
 scanf("%d",&r);

 printf("Enter the Max Matrix:");
 for(i=0;i<p;i++)
 {
    printf("\nFor Process %d:",i+1);
   for(j=0;j<r;j++)
    scanf("%d",&Max[i][j]);
 }

 printf("Enter the Allocation Matrix:");
 for(i=0;i<p;i++)
 {
  printf("\nFor Process %d:",i+1);
   for(j=0;j<r;j++)
    scanf("%d",&alloc[i][j]);
 }


 printf("Need Matrix :\n");
 for(i=0;i<p;i++)
```

```
 {
   for(j=0;j<r;j++){
   need[i][j]=Max[i][j] - alloc[i][j];
  printf("%d\t",need[i][j]);
 }
 printf("\n");
 }
 return 0;
}
```

Slip 19

Q.1 Write a program to create a child process using fork().The parent should goto sleep state and child process should begin its execution. In the child process, use execl() to execute the “ls” command

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
main()
{
   int pid;
   pid = fork();
   if (pid == 0)
   {
      printf("Child process is running\n ");
      execl("/bin/ls", "ls", NULL);
   }
   else
   {
      printf("Child Process is terminated\n");
      sleep(2);
   }
   return 0;
}
```

Slip 20

Q.1 Write a program to create a child process using fork().The parent should goto sleep state

and child process should begin its execution. In the child process, use execl() to execute the “ls” command

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
main()
{
   int pid;
   pid = fork();
   if (pid == 0)
   {
      printf("Child process is running\n ");
      execl("/bin/ls", "ls", NULL);
   }
   else
   {
      printf("Child Process is terminated\n");
      sleep(2);
   }
   return 0;
}
```