1. Write a C program to illustrate the concept of orphan process. Parent process creates a

child and terminates before child has finished its task. So child process becomes orphan

process. (Use fork(), sleep(), getpid(), getppid()). [10]

1. Write a simulation program to implement demand paging using LFU page replacement algorithm. Assume the memory of “n” frames. Show the contents of page after every page replacement in a frame and it at end we told number of page faults accordingly. Input the following page reference string.

Page reference string : 3,4,5,4,3,4,7,2,4,5,6,7,2,4,6 [20]

1. Viva [05]
2. Implement the C Program to create a child process using fork(), display parent and child

process id. Child process will display the message “I am Child Process” and the parent

process should display “I am Parent Process”. [10]

1. Write a simulation program to implement Round Robin CPU scheduling algorithm for the given time quantum as input. Also accept the number of processes and arrival time and CPU burst time for each process as input. The output should give the Gant Chart, turnaround time and waiting time for each process. Also display the average turnaround time and average waiting time. [20]
2. Viva [05]
3. 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]
4. Write a simulation program to implement demand paging using LRU (Using Counter) page replacement algorithm. Assume the memory of “n” frames. Show the contents of page after every page replacement in a frame and at end show the total number of page faults accordingly. Input the following Page Reference String.

Page Reference String :12,15,12,18,6,8,11,12,19,12,6,8,12,15,19,8 [20]

1. Viva [05]
2. 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]

1. Write a program to implement a toy shell (Command Interpreter). It has its own prompt say “MyShell $”. Any normal shell command is executed from this shell (MyShell$) by starting a child process to execute the system program corresponding to the command. It should additionally interpret the following commands:

list f dirname : To print names of all the files in current directory

list n dirname : To print the number of all entries in the current directory.

list i dirname : To print names and inodes of the files in the current directory. [20]

1. Viva [05]
2. Write a C program that behaves like a shell which displays the command prompt ‘myshell$’. It accepts the command, tokenize the command line and execute it by creating the child process. Also implement the additional command ‘count’ as

myshell$ count c filename: It will display the number of characters in given file [10]

1. Write the program to simulate FCFS CPU-scheduling. The arrival time and first CPU burst for different n number of processes should be input to the algorithm. Assume that the fixed IO waiting time (2 units). The next CPU-burst should be generated randomly. The output should give Gantt chart, turnaround time and waiting time for each process. Also find the average waiting time and turnaround time. [20]
2. Viva [05]
3. Write a C program that behaves like a shell which displays the command prompt ‘myshell$’. It accepts the command, tokenize the command line and execute it by creating the child process. Also implement the additional command ‘count’ as

myshell$ count w filename: It will display the number of words in given file [10]

1. Write a simulation program to implement demand paging using MFU page replacement algorithm. Assume the memory of “n” frame. Show the contents of page after every page replacement in a frame and at end show the total number of page faults accordingly. Input the following Page Reference String.

Page Replacement String: 8, 5, 7, 8, 5, 7, 2, 3, 7, 3, 5, 9, 4, 6, 2 [20]

1. Viva [05]
2. Write a C program that behaves like a shell which displays the command prompt ‘myshell$’. It accepts the command, tokenize the command line and execute it by creating the child process. Also implement the additional command ‘count’ as

myshell$ count l filename: It will display the number of lines in given file [10]

1. Write the simulation program to implement demand paging using FIFO page replacement algorithm. Assume the memory of “n” frames. Show the contents of page after every page replacement in a frame and at end show the total number of page faults accordingly. Input the following Page Reference String.

Reference String: 3,4,5,6,3,4,7,3,4,5,6,7,2,4,6 [20]

1. Viva [05]
2. 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]
3. Write a C program that behaves like a shell which displays the command prompt ‘myshell$’. It accepts the command, tokenize the command line and execute it by creating the child process. Also implement the additional command ‘typeline’ as

myshell$ typeline n filename: It will display first n lines of the file.

myshell$ typeline -n filename: It will display last n lines of the file.

myshell$ typeline a filename: It will display all the lines of the file. [20]

1. Viva [05]

1. Write a program to implement a toy shell (Command Interpreter). It has its own prompt say “MyShell $”. Any normal shell command is executed from this shell (MyShell$) by starting a child process to execute the system program corresponding to the command. It should additionally interpret the following commands:

list n dirname : To print the number of all entries in the current directory [10]

1. Write a simulation program to implement demand paging using FIFO page replacement algorithm. Assume the memory of “n” frames. Show the contents of page after every page replacement in a frame and at end show the total number of page faults accordingly. Input the following Page Reference String.

Page Reference String : 3,4,5,4,3,4,7,2,4,5,6,7,2,4,6 [20]

1. Viva [05]
2. Implement the C Program to create a child process using fork(), display parent and child

process id. Child process will display the message “I am Child Process” and the parent

process should display “I am Parent Process”. [10]

1. Write a simulation program to implement Pre-emptive Priority CPU scheduling algorithm. Accept the number of processes, arrival time, CPU burst time and priority for each process as input. Priorities should in High to Low order (1 is High). The output should give the Gantt chart, turnaround time and waiting time for each process. Also display the average turnaround time and average waiting time. [20]
2. Viva [05]
3. Write a program to implement a toy shell (Command Interpreter). It has its own prompt say “MyShell $”. Any normal shell command is executed from this shell (MyShell$) by starting a child process to execute the system program corresponding to the command. It should additionally interpret the following commands:

list f dirname : To print names of all the files in current directory [10]

1. Implement the C program to accept n integers to be sorted. Main function creates child process using fork system call. Parent process sorts the integers using bubble sort and waits for child process using wait system call. Child process sorts the integers using insertion sort [20]
2. Viva [05]
3. Write a C program that behaves like a shell which displays the command prompt ‘myshell$’. It accepts the command, tokenize the command line and execute it by creating the child process. Also implement the additional command ‘count’ as

myshell$ count l filename: It will display the number of lines in given file [10]

1. Write a simulation program to implement FCFS CPU scheduling algorithm. Accept the number of processes, arrival time, CPU burst time and priority for each process as input. Priorities should in High to Low order (1 is High). The output should give the Gantt chart, turnaround time and waiting time for each process. Also display the average turnaround time and average waiting time. [20]
2. Viva [05]
3. Implement the C Program to create a child process using fork(), display parent and child

process id. Child process will display the message “I am Child Process” and the parent

process should display “I am Parent Process”. [10]

1. Write a program to implement a toy shell (Command Interpreter). It has its own prompt say “MyShell $”. Any normal shell command is executed from this shell (MyShell$) by starting a child process to execute the system program corresponding to the command.

It should additionally interpret the following commands:

search f file name :- To search first occurrence of the pattern in the file

search a file name :- To search all the occurrence of the pattern in the file

search c file name :- To count the number of occurrence of the pattern in the file.

[20]

3. Viva. [05]

1. Write a C program to illustrate the concept of orphan process. Parent process creates a

child and terminates before child has finished its task. So child process becomes orphan

process. (Use fork(), sleep(), getpid(), getppid()). [10]

1. Write a program to implement a toy shell (Command Interpreter). It has its own prompt say “MyShell $”. Any normal shell command is executed from this shell (MyShell$) by starting a child process to execute the system program corresponding to the command.

It should additionally interpret the following commands:

typeline +n filename :- To print first n lines in the file. Typeline –n filename :- To print last n lines in the file.

typeline a filename :- To print all lines in the file. [20]

1. Viva [05]

1. Write a C program to illustrate the concept of orphan process. Parent process creates a

child and terminates before child has finished its task. So child process becomes orphan

process. (Use fork(), sleep(), getpid(), getppid()). [10]

1. Write a simulation program to implement FCFS CPU-scheduling algorithm. Accept the number of Processes as input. Also accept arrival time and CPU burst time for each process as input. The output should give the Gantt chart, turnaround time and waiting time for each process. Also display the average turnaround time and average waiting time. [20]
2. Viva [05]