Set-1

1)Explain about

LS:

ls Command in Linux

Last Updated : 09 Jun, 2023

ls is a Linux shell command that lists directory contents of files and directories. It provides valuable information about files, directories, and their attributes.

Syntax of `ls` command in Linux

```
ls [option] [file/directory]
```

'ls' will display the contents of the current directory. By default, 'ls' lists files and directories in alphabetical order.

B) DU:

The `du` command in Linux is a powerful utility that allows users to analyze and report on disk usage within directories and files.

Example: du

```
/home/mandeep/test
```

Output:

```
44    /home/mandeep/test/data
2012    /home/mandeep/test/system design
24    /home/mandeep/test/table/sample_table/tree
28    /home/mandeep/test/table/sample_table
32    /home/mandeep/test/table
100104    /home/mandeep/test
```

2)WRITE A C PROGRAM TO SIMULATE INDEXED FILE ALLOCATION

#include <stdio.h>

```
int main() {     int n, m[20], i, j, sb[20],
b[20][20], x;     printf("Enter no. of
files: ");     scanf("%d", &n);
   for (i = 0; i < n; i++) {         printf("Enter index
block of file%d: ", i + 1);         scanf("%d",
&sb[i]);         printf("Enter length of file%d: ", i +
1);         scanf("%d", &m[i]);         printf("Enter
blocks of file%d: ", i + 1);         for (j = 0; j <
m[i]; j++) {             scanf("%d", &b[i][j]);
       }
   }
   printf("File\tIndex\tLength\n");     for (i = 0;
i < n; i++) {         printf("%d\t%d\t%d\n", i +
1, sb[i], m[i]);
   }
   printf("Enter file name (number): ");
scanf("%d", &x);     if (x > 0 && x <=
n) {         printf("File name is: %d", x);
printf("Index is: %d", sb[x - 1]);
printf("Blocks occupied are:");
for (j = 0; j < m[x - 1]; j++) {
printf("%4d", b[x - 1][j]);
       }         printf("\n");     } else {
printf("Invalid file name.\n");
   }
return 0;
}
```

3)a)bit stuffing :

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

void main() {
   int a[15];     int i, j, k, n, c = 0, pos =
0;     printf("\nEnter the number of bits:
");     scanf("%d", &n);     for (i = 0; i <
n; i++)        scanf("%d", &a[i]);     for
(i = 0; i < n; i++) {        if (a[i] == 1) {
c++;            if (c == 5) {                pos
= i + 1;                c = 0;                for (j
= n; j >= pos; j--) {                    k = j +
1;                    a[k] = a[j];                }
a[pos] = 0;                n = n + 1;
          }
} else
c = 0;
   }     printf("\nData after
stuffing: ");
printf("011111110");     for (i = 0;
i < n; i++) {        printf("%d",
a[i]);

   }
   printf("011111110");
getch();
}
```

b) character stiffing

```
#include <stdio.h>

#include <conio.h>

#include <string.h>


int main() {     char a[100], b[100], c[100];     int i, j =

0, n, k = 0;     printf("Enter the data: ");     gets(a);     n

= strlen(a);     for (i = 0; i < n; i++) {         if ((a[i] == 'd'

&& a[i+1] == 'l' && a[i+2] == 'e') ||             (a[i] == 'e'

&& a[i+1] == 's' && a[i+2] == 'c')) {             c[k++] =

'e';             c[k++] = 's';             c[k++] = 'c';

        }         c[k++]

= a[i];

    }     c[k] = '\0';

b[j] = '\0';

printf("%s\n", b);

printf("DLESTX");

printf("%s", c);

printf("DLEETX");

    return 0;

}
```

4) First-Come, First-Served (FCFS) scheduling

```
#include <stdio.h>


void main() {     int pid[10], bt[10], wt[10], tat[10], n, twt

= 0, ttat = 0, i;     float awt, atat;     printf("Enter number

of processes: ");     scanf("%d", &n);     printf("Enter
```

```
burst times: ");     for (i = 0; i < n; i++)         scanf("%d",
&bt[i]);     wt[0] = 0;     tat[0] = bt[0];     for (i = 1; i < n;
i++) {         wt[i] = tat[i - 1];         tat[i] = bt[i] + wt[i];     }
for (i = 0; i < n; i++)     {         ttat += tat[i];         twt +=
wt[i]; }      printf("\nPID \tBT \tWT \tTAT");     for (i = 0;
i < n; i++)         printf("\n%d\t%d\t%d\t%d", i + 1, bt[i],
wt[i], tat[i]);     awt = (float)twt / n;     atat = (float)ttat / n;
printf("\nAverage Waiting Time = %f", awt);
printf("\nAverage Turnaround Time = %f", atat);
}
```

Set-2

1)a)DATE COMMMAND:

The "**date**" command in Linux is a simple but powerful tool used to display the current date and time, as well as set the system date and time. This command is extremely useful for troubleshooting and system administration tasks, and is a vital tool in understanding any Linux user.

```
$ date
Tue Jan 25 14:20:34 EST 2022
```

B)CHMOD:

Linux chmod command is used to change the access permissions of files and directories. It stands for **change mode**. It can not change the permission of symbolic links. Even, it ignores the symbolic links come across recursive directory traversal. *Examples of Using the Symbolic mode:*

- Read, write and execute permissions to the file owner:

```
chmod u+rwx [file_name]
```

- Remove write permission for the group and others:

```
chmod go-w [file_name]
```

2) C PAROGRAM TO SIMLUTATE PAGGING

```
#include <stdio.h>

void main() {     int i, j, temp, framearr[20], pages, pageno, frames, memsize, log, pagesize,
prosize, base;     printf("Enter the Process size: ");     scanf("%d", &prosize);
printf("Enter the main memory size: ");     scanf("%d", &memsize);     printf("Enter the
page size: ");     scanf("%d", &pagesize);     pages = prosize / pagesize;
   printf("\nThe process is divided into %d pages", pages);     frames =
memsize / pagesize;     printf("\n\nThe main memory is divided into %d
frames\n", frames);     for (i = 0; i < frames; i++)         framearr[i] = -1; /*
Initializing array elements with -1 */     for (i = 0; i < pages; i++) {
pos: printf("\nEnter the frame number of page %d: ", i);
scanf("%d", &temp); /* storing frameno in temporary variable */         if
(temp >= frames) { /* checking whether frameno is valid or not */
printf("\n\t**** Invalid frame number ****\n");             goto pos;
       }
       /* storing pageno (i.e 'i') in framearr at framno (i.e temp) index */
for (j = 0; j < frames; j++)             if (temp == j)
framearr[temp] = i;
   }
   printf("\n\nFrame no\tPage no\tValidation Bit\n------\t------\t----------");
for (i = 0; i < frames; i++) {         printf("\n %d \t %2d \t", i, framearr[i]);
if (framearr[i] == -1)             printf(" 0");         else             printf(" 1");
   }
   printf("\nEnter the logical address: ");
scanf("%d", &log);     printf("\nEnter
the base address: ");     scanf("%d",
&base);     pageno = log / pagesize;
```

```
   for (i = 0; i < frames; i++)        if (framearr[i] ==
pageno) {            temp = i;            j = log % pagesize;
/* here 'j' is displacement */
        temp = base + (temp * pagesize) + j; // lhs 'temp' is physical address rhs and 'temp' is
frame num
        printf("\nPhysical address is : %d", temp);
      }
}
```

3) code for crc

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

void main() {     int i, j, keylen, msglen;     char input[100], key[30],
temp[30], quot[100], rem[30], key1[30];

   printf("Enter Data: ");
gets(input);     printf("Enter
Key: ");     gets(key);

   keylen = strlen(key);
msglen = strlen(input);
strcpy(key1, key);

   for (i = 0; i < keylen - 1; i++) {
input[msglen + i] = '0';
   }
```

```
    for (i = 0; i < keylen; i++) {
temp[i] = input[i];
    }

    for (i = 0; i < msglen; i++) {
quot[i] = temp[0];

        if (quot[i] == '0') {              for
(j = 0; j < keylen; j++) {
key[j] = '0';
            }         } else {              for (j
= 0; j < keylen; j++) {
key[j] = key1[j];
            }
        }

        for (j = keylen - 1; j > 0; j--) {
if (temp[j] == key[j]) {
rem[j - 1] = '0';              } else {
rem[j - 1] = '1';
            }
        }

        rem[keylen - 1] = input[i + keylen];
strcpy(temp, rem);
    }

    strcpy(rem, temp);
```

```
   printf("\nQuotient is ");
for (i = 0; i < msglen; i++) {
printf("%c", quot[i]);
   }

   printf("\nRemainder is ");
for (i = 0; i < keylen - 1; i++) {
printf("%c", rem[i]);
   }

   printf("\nFinal data is: ");
for (i = 0; i < msglen; i++) {
printf("%c", input[i]);
   }

   for (i = 0; i < keylen - 1; i++) {
printf("%c", rem[i]);
   }

   getch();
}
```

4)sjf code

```
#include<stdio.h>

void main() {     int pid[10], bt[10], wt[10], tat[10], n, twt = 0,
ttat = 0, i, j, t;     float awt, atat;
```

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

   printf("\nEnter burst times:\n");
for (i = 0; i < n; i++) {
scanf("%d", &bt[i]);
   }

   printf("\nEnter PID:\n");
for (i = 0; i < n; i++) {
scanf("%d", &pid[i]);
   }
   // Sorting processes based on burst time using Bubble Sort
for (i = 0; i < n; i++) {         for (j = i + 1; j < n; j++) {
if (bt[i] > bt[j]) {                t = bt[i];                bt[i] = bt[j];
bt[j] = t;                t = pid[i];                pid[i] = pid[j];
pid[j] = t;
           }
       }
   }

   wt[0] = 0;
tat[0] = bt[0];     //
Calculating
waiting time and
turnaround time
for (i = 1; i < n;
i++) {         wt[i]
```

```
= tat[i - 1];
tat[i] = bt[i] +
wt[i];
    }
    // Calculating total waiting time and total turnaround time
for (i = 0; i < n; i++) {        ttat += tat[i];        twt += wt[i];
    }

    printf("\nPID \tBT \tWT \tTAT\n");     for (i = 0; i < n; i++)
{       printf("%d \t%d \t%d \t%d\n", pid[i], bt[i], wt[i], tat[i]);
    }

    awt = (float)twt / n;
atat = (float)ttat / n;

    printf("\nAvg. Waiting Time = %f", awt);
printf("\nAvg. Turnaround Time = %f\n", atat);
}
```

Set3

1***rm***

a) delete - `rm {file-name} for single file`

b)ps Syntax of `ps` Command in Linux

The `ps` command provides a snapshot of the current processes on your system.

The basic syntax is as follows:

```
ps [options]
```

c)df

Syntax of `df` command in Linux

The basic syntax of `df` is:

```
df [options] [filesystems]
```

Here,

- `options`: These are optional flags that modify the output of the command. We'll discuss some important ones later.
- `filesystems`: You can specify specific filesystems (mount points) to check their usage instead of getting information for all mounted drives.

If no file name is given, it displays the space available on all currently mounted file systems. **For example:**

```
df
```

2)sjf code

```
a#include<stdio.h>


void main() {     int pid[10], bt[10], wt[10], tat[10], n, twt = 0,
ttat = 0, i, j, t;     float awt, atat;


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


   printf("\nEnter burst times:\n");
for (i = 0; i < n; i++) {
scanf("%d", &bt[i]);
   }


   printf("\nEnter PID:\n");
for (i = 0; i < n; i++) {
scanf("%d", &pid[i]);
   }
```

```
    // Sorting processes based on burst time using Bubble Sort
for (i = 0; i < n; i++) {         for (j = i + 1; j < n; j++) {
if (bt[i] > bt[j]) {                t = bt[i];                bt[i] = bt[j];
bt[j] = t;                t = pid[i];                pid[i] = pid[j];
pid[j] = t;
            }
        }
    }

    wt[0] = 0;
tat[0] = bt[0];

    // Calculating waiting time and turnaround time
for (i = 1; i < n; i++) {         wt[i] = tat[i - 1];
tat[i] = bt[i] + wt[i];
    }
    // Calculating total waiting time and total turnaround time
for (i = 0; i < n; i++) {         ttat += tat[i];         twt += wt[i];
    }

    printf("\nPID \tBT \tWT \tTAT\n");     for (i = 0; i < n; i++)
{       printf("%d \t%d \t%d \t%d\n", pid[i], bt[i], wt[i], tat[i]);
    }

    awt = (float)twt / n;
atat = (float)ttat / n;

    printf("\nAvg. Waiting Time = %f", awt);
printf("\nAvg. Turnaround Time = %f\n", atat);
```

```
}
```

3) code for sequential file allocation

```
#include<stdio.h>

void main() {     int n, i, j, b[20], sb[20],
t[20], x, c[20][20];     printf("Enter number
of files: ");     scanf("%d", &n);

   for(i = 0; i < n; i++) {         printf("Enter number of blocks
occupied by file %d: ", i + 1);         scanf("%d", &b[i]);
printf("Enter the starting block of file %d: ", i + 1);
scanf("%d", &sb[i]);         t[i] = sb[i];

       for(j = 0; j < b[i]; j++)
c[i][j] = sb[i]++;
   }

   printf("\nFilename\tStart block\tLength\n");
for(i = 0; i < n; i++)
printf("%d\t\t%d\t\t%d\n", i + 1, t[i], b[i]);

   printf("\nEnter file name: ");
scanf("%d", &x);     printf("\nFile
name is: %d", x);     printf("\nLength
is: %d", b[x - 1]);     printf("\nBlocks
occupied:");
```

```
    for(i = 0; i < b[x - 1]; i++)
printf("%4d", c[x - 1][i]);
}
```

4) impletation of stop and wait protocol

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

void main() {     int i, j, noframes,
x1 = 10, x2, x;

    // Initialize random seed
srand(time(NULL));

    // Generate number of frames
    noframes = rand() % 200;
    noframes = noframes / 8;

    printf("\nNumber of frames is %d", noframes);
        i
= 1;     j
= 1;

    while (noframes > 0) {
printf("\nSending frame %d", i);
```

```
        // Simulate random delay          x = rand() %
10;         if (x % 2 == 0) {              for (x2 = 1; x2
< 2; x2++) {                  printf("Waiting for %d
seconds\n", x2);
                // Simulate waiting with sleep (not standard in C, often implemented in libraries)
                // For standard C, you can use a different approach or use platform-specific
functions.
            }
            printf("\nSending frame %d", i);
x = rand() % 10;
        }

        printf("\nACK for frame %d", j);
noframes -= 1;

i++;
j++; }

    printf("\nEnd of stop and wait protocol");
getch();  // Waits for a key press before exiting }
```

Set-4

1)pwd :

Syntax of `pwd` command in Linux

The basic syntax of the ‘pwd’ command is `pwd [OPTIONS]`

b) Syntax of who Command in Linux `who [options] [filename]`

Practical Examples of who Command

1. The who command displays the following information for each user currently logged in to the system if no option is provided :

1. Login name of the users
2. Terminal line numbers
3. Login time of the users into the system 4. The remote host name of the user

```
hduser@mahesh-Inspiron-3543:~$ who hduser
tty7         2018-03-18 19:08 (:0)
```

2) Round Robin scheduling

```
#include<stdio.h>


void main() {     int ts, bt1[10], wt[10], tat[10], i, j = 0, n, bt[10], ttat = 0,
twt = 0, tot = 0;     float awt, atat;


   printf("Enter the number of Processes: ");
scanf("%d", &n);


   printf("\nEnter the Timeslice: ");
   scanf("%d", &ts);


   printf("\nEnter the Burst Time for each process:\n");
for (i = 1; i <= n; i++) {         printf("P[%d]: ", i);
scanf("%d", &bt1[i]);         bt[i] = bt1[i];
   }


   while (j < n) {         for (i
= 1; i <= n; i++) {             if
(bt[i] > 0) {                 if
(bt[i] >= ts) {
tot += ts;                     bt[i] -
```

```
= ts;                } else {
tot += bt[i];
bt[i] = 0;
            }
if (bt[i] == 0) {
j++;                    tat[i] =
tot;
            }
          }
       }
   }

   for (i = 1; i <= n; i++) {
wt[i] = tat[i] - bt1[i];         twt
+= wt[i];         ttat += tat[i];
   }

   awt = (float)twt / n;
atat = (float)ttat / n;
       printf("\nPID \tBT \tWT \tTAT\n");    for (i = 1; i <=
n; i++) {            printf("%d \t%d \t%d \t%d\n", i, bt1[i],
wt[i], tat[i]);
   }
      printf("\nThe average Waiting Time = %f", awt);
printf("\nThe average Turnaround Time = %f\n", atat);
}
```

3)LRU PAGE REPLACEMET ALG

```
#include<stdio.h>

void main() {     int i, j, l, max, n, a[50], frame[10], flag, fno, k, avail,
pagefault = 0, lru[10];

   printf("\nEnter the number of Frames : ");
scanf("%d", &fno);

   printf("\nEnter number of reference string :");
scanf("%d", &n);

   printf("\nEnter the Reference string :\n");
for(i = 0; i < n; i++)         scanf("%d",
&a[i]);

   for(i = 0; i < fno; i++) {
frame[i] = -1;         lru[i]
= 0;
   }

   printf("\nLRU Page Replacement Algorithm\n\nThe given reference string is:\n\n");
for(i = 0; i < n; i++) {         printf(" %d ", a[i]);
   }
printf("\n");
    j = 0;     for(i = 0; i < n; i++) {
max = 0;         flag = 0;
printf("\nReference No %d-> ", a[i]);
```

```
avail = 0;          for(k = 0; k < fno; k++) {
if(frame[k] == a[i]) {                  avail = 1;
lru[k] = 0;                 break;
          }         }         if(avail == 1)
{             for(k = 0; k < fno; k++) {
if(frame[k] != -1)
++lru[k];                 max = 0;
for(k = 1; k < fno; k++) {
if(lru[k] > lru[max])
max = k;                  }                 j =
max;
          }         }         if(avail ==
0) {              lru[j] = 0;
frame[j] = a[i];              for(k = 0;
k < fno; k++) {
if(frame[k] != -1)
++lru[k];                 else {
j = k;                    flag = 1;
break;
               }            }            if(flag
== 0) {                 max = 0;
for(k = 1; k < fno; k++) {
if(lru[k] > lru[max])
max = k;                  }                 j =
max;
          }             pagefault++;
for(k = 0; k < fno; k++) {
if(frame[k] != -1)
```

```
printf(" %2d", frame[k]);            }
}        printf("\n");
   }
   printf("\nPage Fault Is %d\n", pagefault);
}
```

4)SLIDING WINDOW PROTOCOL

```
#include<stdio.h>

int main() {     int w, i, f,
frames[50];

   printf("Enter window size: ");
scanf("%d", &w);

   printf("\nEnter number of frames to transmit: ");
scanf("%d", &f);

   printf("\nEnter %d frames: ", f);
for(i = 1; i <= f; i++) {
scanf("%d", &frames[i]);
   }

   printf("\nWith sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)\n\n");
   printf("After sending %d frames at each stage sender waits for acknowledgement sent by the receiver\n\n", w);
```

```
   for(i = 1; i <= f; i++) {
      if(i % w == 0) {            printf("%d\n", frames[i]);
printf("Acknowledgement of above frames sent is received by sender\n\n");
      } else {            printf("%d
", frames[i]);
      }
   }

   if(f % w != 0) {         printf("\nAcknowledgement of above frames sent is
received by sender\n");
   }

   return 0;
}
```

SET-5

1)factorial problem:

2)LINKED FILE

```
#include<stdio.h>

struct file {     char
fname[10];     int start,
size, block[10];
};

int main() {
struct file f[10];
int i, j, n;

   printf("Enter no. of files:");
scanf("%d", &n);

   for(i = 0; i < n; i++) {
printf("Enter file name:");
scanf("%s", f[i].fname);

      printf("Enter starting block:");
scanf("%d", &f[i].start);
f[i].block[0] = f[i].start;

      printf("Enter no. of blocks:");
scanf("%d", &f[i].size);
```

```
printf("Enter block numbers:");
for(j = 1; j < f[i].size; j++) {
scanf("%d", &f[i].block[j]);
      }
   }

   printf("File\tstart\tsize\tblock\n");     for(i = 0; i < n;
i++) {        printf("%s\t%d\t%d\t", f[i].fname, f[i].start,
f[i].size);        for(j = 0; j < f[i].size - 1; j++) {
printf("%d--->", f[i].block[j]);
      }
      printf("%d\n", f[i].block[j]);
   }

   return 0;
}
```

3)OPTIML PAGE REPLACE METN

```
#include<stdio.h>

int main() {     int i, j, l, min, flag1, n, temp, frame[10], fno, k, avail, pagefault = 0,
opt[10], a[50];

   printf("\nEnter the number of Frames : ");
scanf("%d", &fno);

   printf("\nEnter number of reference string : ");
scanf("%d", &n);
```

```
   printf("\nEnter the Reference string :\n");
   for(i = 0; i < n; i++) {
scanf("%d", &a[i]);
   }

   for(i = 0; i < fno; i++) {
frame[i] = -1;        opt[i]
= 0;
   }

   printf("\nOptimal Page Replacement Algorithm\n\nThe given reference string is:\n\n");
for(i = 0; i < n; i++) {        printf(" %d ", a[i]);
   }
printf("\n");
    j = 0;     for(i = 0; i < n; i++) {
flag = 0;        flag1 = 0;
printf("\nReference No %d-> ", a[i]);
avail = 0;

      for(k = 0; k < fno; k++) {
if(frame[k] == a[i]) {
avail = 1;                break;
         }
      }

      if(avail == 0) {
         temp = frame[j];
frame[j] = a[i];
```

```
        for(k = 0; k < fno; k++) {
if(frame[k] == -1) {
                j = k;
flag = 1;
break;
            }
        }

        if(flag == 0) {
for(k = 0; k < fno; k++) {
opt[k] = 0;                 for(l = i; l <
n; l++) {                   if(frame[k]
== a[l]) {                      flag1 =
1;                          break;
                    }
}                   if(flag1 == 1)
{                       opt[k] = l -
i;                  } else {
opt[k] = -1;
break;
                }
            }
            min = 0;            for(k
= 0; k < fno; k++) {
if(opt[k] < opt[min] && opt[k] != -
1) {                    min = k;
} else if(opt[k] == -1) {
```

```
min = k;                    frame[j] =
temp;                     frame[k] =
a[i];                   break;
                }
}               j =
min;
          }
       }

       pagefault++;        for(k = 0; k
< fno; k++) {           if(frame[k]
!= -1) {              printf(" %2d",
frame[k]);
          }        }
printf("\n");
   }

   printf("\nPage Fault Is %d", pagefault);
return 0;
}
```

4)SHORTEST PATH

READ LAB RECRD

SET-6

1)a)cp only copies

Mv will delete the source file

b)cat>file name rp

c)df-disk free also known as `df`, which is a powerful utility that provides valuable information on disk space utilization. The **df** command displays information about file system disk space usage on the mounted file system

df- for all df

filename

2)PRIORITY

```
#include<stdio.h>


void main()
{     int pid[10], bt[10], pr[10], wt[10], tat[10], n, twt = 0, ttat = 0, i, j,
t;     float awt, atat;


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


   printf("\nEnter burst times:");
for (i = 0; i < n; i++)
scanf("%d", &bt[i]);


   printf("\nEnter PID:");
for (i = 0; i < n; i++)
scanf("%d", &pid[i]);
printf("\nEnter Priorities:");
```

```
for (i = 0; i < n; i++)
scanf("%d", &pr[i]);


   // Sorting processes based on priority (ascending order)
for (i = 0; i < n; i++)
   {        for (j = i + 1; j < n;
j++)
       {            if (pr[i]
> pr[j])
           {               t =
pr[i];               pr[i] =
pr[j];               pr[j] =
t;
               t = bt[i];
bt[i] = bt[j];
bt[j] = t;
               t = pid[i];
pid[i] = pid[j];
pid[j] = t;
           }
       }
   }
   // Calculating waiting time and turnaround time
wt[0] = 0;     tat[0] = bt[0];     for (i = 1; i < n;
i++)
   {        wt[i] = tat[i -
1];        tat[i] = bt[i] +
wt[i];
```

```
    }

    // Calculating total waiting time and total turnaround time
for (i = 0; i < n; i++)
    {       ttat +=
tat[i];       twt +=
wt[i];
    }
    // Printing the results          printf("\n PID \t
PRIORITY \t BT \t WT \t TAT");     for (i = 0; i < n;
i++)
    {       printf("\n %d \t\t %d \t\t %d \t\t %d \t\t %d", pid[i], pr[i], bt[i], wt[i],
tat[i]);
    }

    // Calculating and printing average waiting time and average turnaround time
awt = (float)twt / n;     atat = (float)ttat / n;     printf("\n\nAvg. Waiting Time =
%f", awt);     printf("\nAvg. Turnaround Time = %f\n", atat);
}
```

3)BORDCSAT TREE

```
#include<stdio.h>

int a[10][10], n;

void adj(int k);

int main() {
    int i, j, root;
```

```
    printf("Enter number of nodes: ");
scanf("%d", &n);

    printf("Enter adjacency matrix:\n");     for (i = 1; i <= n; i++) {
for (j = 1; j <= n; j++) {             printf("Is there a connection between
%d --> %d (1/0): ", i, j);             scanf("%d", &a[i][j]);
        }
    }

    printf("Enter root node: ");
scanf("%d", &root);

    adj(root);

    return 0;
}

void adj(int k) {
    int i, j;

    printf("Adjacent nodes of root node %d:\n", k);

    // Print adjacent nodes
    for (j = 1; j <= n; j++) {         if
(a[k][j] == 1 || a[j][k] == 1) {
printf("%d\t", j);
```

```
    }   }
printf("\n");

  // Print non-adjacent nodes   printf("Non-adjacent
nodes of root node %d:\n", k);   for (i = 1; i <= n;
i++) {      if (i != k && a[k][i] == 0 && a[i][k] ==
0) {          printf("%d\t", i);
    }   }
printf("\n"); }
```

4)LRU

```
#include<stdio.h>

void main() {   int i, j, l, max, n, a[50], frame[10], flag, fno, k, avail,
pagefault = 0, lru[10];

  printf("\nEnter the number of Frames : ");
scanf("%d", &fno);

  printf("\nEnter number of reference string :");
scanf("%d", &n);

  printf("\nEnter the Reference string :\n");
for(i = 0; i < n; i++)
    scanf("%d", &a[i]);

  for(i = 0; i < fno; i++) {
frame[i] = -1;      lru[i]
= 0;
```

```
    }

    printf("\nLRU Page Replacement Algorithm\n\nThe given reference string is:\n\n");
for(i = 0; i < n; i++) {        printf(" %d ", a[i]);
    }
printf("\n");
    j = 0;     for(i = 0; i < n; i++) {
max = 0;        flag = 0;
printf("\nReference No %d-> ", a[i]);
avail = 0;        for(k = 0; k < fno; k++) {
if(frame[k] == a[i]) {                avail = 1;
lru[k] = 0;                break;
          }        }        if(avail ==
1) {            for(k = 0; k < fno;
k++) {                if(frame[k] != -
1)
                ++lru[k];
            max = 0;                for(k
= 1; k < fno; k++) {
if(lru[k] > lru[max])
max = k;                }                j =
max;
          }        }        if(avail ==
0) {            lru[j] = 0;
frame[j] = a[i];            for(k = 0;
k < fno; k++) {
if(frame[k] != -1)
++lru[k];                else {
```

```
j = k;                    flag = 1;
break;
            }          }          if(flag
== 0) {              max = 0;
for(k = 1; k < fno; k++) {
if(lru[k] > lru[max])
max = k;              }              j =
max;
        }          pagefault++;
for(k = 0; k < fno; k++) {
if(frame[k] != -1)
printf(" %2d", frame[k]);
        }      }
printf("\n");
   }
   printf("\nPage Fault Is %d\n", pagefault);
}
```