Practical 1 : File handling operations

```
#include<stdio.h>
#include <stdlib.h>
int main()
{
FILE *fp; /* file pointer*/char
fName[20];
printf("\nEnter file name to create :");
scanf("%s",fName);
fp=fopen(fName,"w");

if(fp==NULL)
{
printf("File does not created!!!");exit(0);
}
printf("File created successfully.");

putc('V',fp);
putc('I',fp);
putc('R',fp);
putc('A',fp);
putc('J',fp);
printf("\nData written successfully.");
fclose(fp);
fp=fopen(fName,"r");
if(fp==NULL)
{
printf("\nCan't open file!!!");
exit(0);
}
printf("Contents of file is :\n");
printf("%c",getc(fp));
printf("%c",getc(fp));
printf("%c",getc(fp));
printf("%c",getc(fp));
printf("%c",getc(fp));
printf("%c",getc(fp));
printf("%c",getc(fp));
fclose(fp);
return 0;
}
```

Output:

```
Enter file name to create : Viraj
File created successfully.
Data written successfully.
Contents of file is : VIRAJ
```

Practical 2 : FCFS CPU scheduling algorithm

```
#include<stdio.h>
#include<conio.h>
main()
{
int bt[20], wt[20], tat[20], i, n;
float wtavg, tatavg;
printf("\nEnter the number of processes --
");
scanf("%d", &n);
for(i=0;i<n;i++)
{
printf("\nEnter Burst Time for Process %d
-- ",
i);scanf("%d", &bt[i]);
}
wt[0] = wtavg = 0;
tat[0] = tatavg = bt[0];
for(i=1;i<n;i++)
{
wt[i] = wt[i-1] +bt[i-1];
tat[i] = tat[i-1] +bt[i];
wtavg = wtavg + wt[i];
tatavg = tatavg + tat[i];
}
printf("\t PROCESS \tBURST TIME \t
WAITING TIME\t TURNAROUND
TIME\n");
for(i=0;i<n;i++)
printf("\n\t P%d \t\t %d \t\t %d \t\t %d", i,
bt[i], wt[i], tat[i]);
printf("\nAverage Waiting Time -- %f",
wtavg/n);
printf("\nAverage Turnaround Time -- %f",
tatavg/n);
getch();
}
```

Output :

```
Enter the number of processes -- 4
Enter Burst Time for Process 0 -- 12
Enter Burst Time for Process 1 -- 23
Enter Burst Time for Process 2 – 02
Enter Burst Time for Process 3 -- 10
PROCESS BURST TIME WAITING TIME
TURNAROUND TIME P0 12 0 12 P1 23
12 35 P2 2 35 37 P3 10 37 47
Average Waiting Time -- 21.000000
 Average Turnaround Time -- 32.750000
```

Practical 3 : Producer-Consumer problem using semaphores

```
#include <stdio.h>
#define MAX_BUFFER_SIZE 10
int buffer[MAX_BUFFER_SIZE];
int in = 0;
int out = 0;
void produce() {
   int produce;
   if ((in + 1) % MAX_BUFFER_SIZE ==
out) {
      printf("\nBuffer is Full");
   } else {
      printf("\nEnter the value: ");
      scanf("%d", &produce);
      buffer[in] = produce;
      in = (in + 1) % MAX_BUFFER_SIZE;
   }
}
void consume() {
   if (in == out) {
      printf("\nBuffer is Empty");
   } else {
      int consume = buffer[out];
      printf("\nThe consumed value is %d",
consume);
      out = (out + 1) %
MAX_BUFFER_SIZE;
   }
}
int main() {
   int choice = 0;
   while (choice != 3) {
      printf("\n1. Produce \t 2. Consume \t
3. Exit");
      printf("\nEnter your choice: ");
      scanf("%d", &choice);
      switch (choice) {
         case 1:
            produce();
            break;
         case 2:
            consume();
            break;
         case 3:
            printf("\nExiting...");
            break;
         default:
            printf("\nInvalid choice! Please
enter 1, 2, or 3.");
      }
   }
   return 0;  }
```

Output :

```
1. Produce 2. Consume 3. Exit
Enter your choice: 2
Buffer is Empty
1. Produce 2. Consume 3. Exit
Enter your choice: 1
Enter the value: 18
1. Produce 2. Consume 3. Exit
Enter your choice: 3
```

Practical 4 : Memory Allocation Technique.

```
#include<stdio.h>
#include<conio.h>
#define max 25
void main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp;
static int bf[max],ff[max];
printf("\n\tMemory Management Scheme -
First Fit");
printf("\nEnter the number of blocks:");
scanf("%d",&nb);
printf("Enter the number of files:");
scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n");
for(i=1;i<=nb;i++)
{
printf("Block %d:",i);
scanf("%d",&b[i]);
}
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{
printf("File %d:",i);
scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
{
for(j=1;j<=nb;j++)
{
if(bf[j]!=1)
{
temp=b[j]-
f[i];
if(temp>=0)
{
ff[i]=j
;
break;
}
}
}
frag[i]=temp;
bf[ff[i]]=1;
}
printf("\nFile_no:\tFile_size
:\tBlock_no:\tBlock_size:\tFragement");
for(i=1;i<=nf;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[
i],b[ff[i]],frag[i]);
getch();
}
```

Output :

```
Memory Management Scheme - First Fit
Enter the number of blocks:3
Enter the number of files:4
Enter the size of the blocks:-
Block 1:500
Block 2:800
Block 3:200
Enter the size ofthe files :-
File 1:1000
File 2:300
File 3:800
File 4:600
File_no: File_size : Block_no: Block_size:
Fragement 1 1000 0 7343200 -800 2 300
1 500 200 3 800 2 800 0 4 600 0 7343200
-400
```

Practical 5 : Sequential File allocation strategies

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

struct fileTable {
   char name[20];
   int sb, nob;
} ft[30];

int main() {
   int i, j, n;
   char s[20];

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

   for (i = 0; i < n; i++) {
      printf("\nEnter file name %d: ", i + 1);
      scanf("%s", ft[i].name);

      printf("Enter starting block of file %d:
", i + 1);
      scanf("%d", &ft[i].sb);

      printf("Enter number of blocks in file
%d: ", i + 1);
      scanf("%d", &ft[i].nob);
   }

   printf("\nEnter the file name to be
searched: ");
   scanf("%s", s);

   for (i = 0; i < n; i++) {
      if (strcmp(s, ft[i].name) == 0)
         break;
   }

   if (i == n)
      printf("\nFile Not Found");
   else {
      printf("\nFILE NAME\tSTART
BLOCK\tNO OF BLOCKS\tBLOCKS
OCCUPIED\n");
      printf("%s\t\t%d\t\t%d\t\t", ft[i].name,
ft[i].sb, ft[i].nob);
      for (j = 0; j < ft[i].nob; j++)
         printf("%d, ", ft[i].sb + j);
   }

   return 0;
}
```

Output :

```
Enter no of files :4
Enter file name 1 :dir
Enter starting block of file 1 :0
Enter no of blocks in file 1 :12
Enter file name 2 :lib
Enter starting block of file 2 :4
Enter no of blocks in file 2 :5
Enter file name 3 :list
Enter starting block of file 3:7
Enter no of blocks in file 3 :3
Enter file name 4 :sub
Enter starting block of file 4:7
Enter no of blocks in file 4 :9
Enter the file name to be searched-- lib
FILE NAME START BLOCK NO OF
BLOCKS BLOCKS OCCUPIED lib 4 5 4,
5, 6, 7, 8
```

Practical 6 : Single level directory file organization technique

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

struct {
   char dname[10], fname[10][10];
   int fcnt;
} dir;

int main() {
   int i, ch;
   char f[30];
   dir.fcnt = 0;

   printf("\nEnter name of directory -- ");
   scanf("%s", dir.dname);

   while (1) {
      printf("\n\n1. Create File\t2. Delete
File\t3. Search File\n4. Display Files\t5.
Exit\nEnter your choice -- ");
      scanf("%d", &ch);

      switch (ch) {
         case 1:
            printf("\nEnter the name of the
file -- ");
            scanf("%s", dir.fname[dir.fcnt]);
            dir.fcnt++;
            break;
         case 2:
            printf("\nEnter the name of the
file -- ");
            scanf("%s", f);
            for (i = 0; i < dir.fcnt; i++) {
               if (strcmp(f, dir.fname[i]) == 0)
{
                  printf("File %s is deleted\n",
f);
                  strcpy(dir.fname[i],
dir.fname[dir.fcnt - 1]);
                  dir.fcnt--;
                  break;
               }
            }
            if (i == dir.fcnt)
               printf("File %s not found\n", f);
            break;
         case 3:
            printf("\nEnter the name of the
file -- ");
            scanf("%s", f);
            for (i = 0; i < dir.fcnt; i++) {
               if (strcmp(f, dir.fname[i]) == 0)
{
                  printf("File %s is found\n",
f);
                  break;
               }
            }
            if (i == dir.fcnt)
               printf("File %s not found\n", f);
            break;
         case 4:
            if (dir.fcnt == 0)
               printf("\nDirectory Empty\n");
            else {
               printf("\nThe Files are --\n");
               for (i = 0; i < dir.fcnt; i++)
                  printf("\t%s\n", dir.fname[i]);
            }
            break;
         default:
            exit(0);
      }
   }
   return 0;
}
```

Output :

```
Enter name of directory -- os
1. Create File 2. Delete File 3. Search File
4. Display Files 5. Exit
Enter your choice ----1
Enter the name of the file -- notes
1. Create File 2. Delete File 3. Search File
4. Display Files 5. Exit
Enter your choice ----1
Enter the name of the file -- reference
books
1. Create File 2. Delete File 3. Search File
4. Display Files 5. Exit
```

Practical 7 : FIFO page replacement algorithm

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

int main() {
   int i, j, k, f, pf = 0, count = 0, rs[25],
m[10], n;

   printf("\n Enter the length of reference
string -- ");
   scanf("%d", &n);

   printf("\n Enter the reference string -- ");
   for (i = 0; i < n; i++)
      scanf("%d", &rs[i]);

   printf("\n Enter number of frames -- ");
   scanf("%d", &f);

   for (i = 0; i < f; i++)
      m[i] = -1;

   printf("\n The Page Replacement
Process is -- \n");

   for (i = 0; i < n; i++) {
      for (k = 0; k < f; k++) {
         if (m[k] == rs[i])
            break;
      }

      if (k == f) {
         m[count++] = rs[i];
         pf++;
      }

      for (j = 0; j < f; j++)
         printf("\t%d", m[j]);

      if (k == f)
         printf("\tPF No. %d", pf);

      printf("\n");

      if (count == f)
         count = 0;
   }

   printf("\n The number of Page Faults
using FIFO are %d\n", pf);

   getch();
   return 0;
}
```

Output :

```
Enter the length of reference string -- 5
Enter the reference string -- 12 5 23 6 9
Enter no. of frames -- 4
The Page Replacement Process is --
12 -1 -1 -1  PF No. 1
12 5 -1 -1 PF No. 2
12 5 23 -1 PF No. 3
12 5 23 6 PF No. 4
9 5 23 6 PF No. 5
The number of Page Faults using FIFO
are 5
```

Practical 8 : FCFS disk scheduling algorithm

```
#include<stdio.h>
int main() {
   int queue[20], n, head, i, j, seek = 0,
max, diff;
   float aver;
   printf("Enter the max range of disk: ");
   scanf("%d", &max);
   printf("Enter the size of queue request:
");
   scanf("%d", &n);
   printf("Enter the queue: ");
   for (i = 1; i <= n; i++) {
      scanf("%d", &queue[i]);
   }
   printf("Enter the initial head position: ");
   scanf("%d", &head);
   queue[0] = head;
   for (j = 0; j < n; j++) {
      diff = abs(queue[j + 1] - queue[j]);
      seek += diff;
      printf("Move is from %d to %d with
seek %d\n", queue[j], queue[j + 1], diff);
   }
   printf("Total seek time is %d\n", seek);
   aver = seek / (float)n;
   printf("Average seek time is %f\n",
aver);
   return 0;
}
```

Output :

```
enter the max range of disk 800
enter the size of queue request 4
enter the queue 110 342 432 456
enter the initial head position 80
move is from 80 to 110with seek 30
move is from 110 to 342 with seek 232
move is from 342 to 432 with seek 90
move is from 432 to 456 with seek 24
total seek time is376
avrage seek time is 94.000000
Process returned 30 (0x1E)
execution time : 26.550
```