SYSTEM SOFTWARE LAB

C G Giridhar
CS-5-B
Roll No- 21

CONTENTS

Experiment 1 :

Simulate the following non-preemptive CPU scheduling algorithms to find turnaround time and waiting time.
a) FCFS b) SJF c) Round Robin (pre-emptive) d) Priority

Program :

```
#include<stdio.h>
void main(){
int choice,size,num=1;

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

int order[size],burst[size],arrival[size],turn=0,wait,priority[size];
int pos,temp,small,quant;
printf("Enter your choice : \n 1.FCFS \n 2.SJF \n 3.RR \n 4.Priority ");
scanf("%d",&choice);
printf("\n");

for(int counter=0;counter<size;counter++)
{
        printf("Enter the burst time and arrival time \n");
        scanf("%d %d",&burst[counter],&arrival[counter]);
        order[counter]=num;
        num++;
}
switch(choice){
        case 1:{

                printf("SI \t Bst \t Arr \t  Turn \t Wait \n");
                for(int j=0;j<size;j++)
                {
                        if(j==0)
                                wait=0;
                        else
                                wait+=burst[j-1];
                        turn+=burst[j];
```

```
                        printf("%d \t %d \t %d ms \t %d ms
\n",order[j],burst[j],arrival[j],turn,wait);

                }

                break;
                }

        case 2:{

                //SORTING

                for(int i=0;i<size;i++)
                { for(int j=1;j<size;j++)
                  {
                        if(burst[j]<burst[j-1])
                        { //SWAPPING BURSTS
                        temp=burst[j];
                        burst[j]=burst[j-1];
                        burst[j-1]=temp;
                        //SWAPPING ORDERS
                                temp=order[j];
                                order[j]=order[j-1];
                                order[j-1]=temp;
                        }
                   }
                }

                //TURN & WAIT
                 printf("SI \t Bst \t Arr \t Turn \t Wait \n");
            for(int j=0;j<size;j++)
            {
                 if(j==0)
                        wait=0;
                 else
                        wait+=burst[j-1];
                 turn+=burst[j];
                        printf("%d \t %d \t %d \t %d ms \t %d ms
\n",order[j],burst[j],arrival[j],turn,wait);

            }

                 break;
```

```
}
        case 3:{
                printf("Enter the time slice :");
                scanf("%d",&quant);
                int turn[size],wait[size],stop,counter=0,t=0,total=0;
                int rem_bt[size];
                for(int i=0;i<size;i++)
                {
                        total+=burst[i];
                        wait[i]=0;
                        rem_bt[i]=burst[i];
                }
                while (1)
                {
                         int flag = 1;

                for (int i = 0 ; i < size; i++)
                {

                        if (rem_bt[i] > 0)
                        {
                        flag = 0;

                        if (rem_bt[i] > quant)
                        {
                                        t += quant;

                              rem_bt[i] -= quant;
                        }
                        else
                        {
                        t = t + rem_bt[i];
                         wait[i] = t - burst[i];
                        rem_bt[i] = 0;
                        }
                        }
                        }
                        if(flag == 1)
                                break;
                }
                for(int i=0;i<size;i++)
                        turn[i]=wait[i]+burst[i];
            printf("Sl \t Bst \t Arr \t Wait \t Turn \n");
```

```
            for(int j=0;j<size;j++)
            {
                  printf("%d \t %d \t %d \t %d ms \t %d ms
\n",order[j],burst[j],arrival[j],wait[j],turn[j]);

            }
                  break;
                  }
          case 4:{

                  printf("Enter the priority: ");
                  for(int i=0;i<size;i++)
                  {
                        scanf("%d",&priority[i]);
                  }
          //PRIORITY SORT
             for(int i=0;i<size;i++)
             { for(int j=1;j<size;j++)
               {
                  if(priority[j]>priority[j-1])
                        { //SWAPPING PRIORITY
                        temp=priority[j];
                        priority[j]=priority[j-1];
                        priority[j-1]=temp;
                   //SWAPPING ORDERS
                        temp=order[j];
                        order[j]=order[j-1];
                        order[j-1]=temp;
                  }
               }
             }
           //TURN & WAIT
              printf("SI \t Bst \t Arr \t Prio  \t Turn \t Wait \n");
             for(int j=0;j<size;j++)
             {
                  if(j==0)
                        wait=0;
                  else
                        wait+=burst[j-1];
                  turn+=burst[j];
                  printf("%d \t %d \t %d \t %d \t  %d ms \t %d ms
\n",order[j],burst[j],arrival[j],priority[j],turn,wait);
                  }
```

```
                break;
                }
                default :{printf("NO Valid Option");break;}
}
}
```

Output :

a) FCFS

Enter the number of processes : 5

Enter your choice :
1.FCFS
2.SJF
3.RR
4.Priority 1

Enter the burst time and arrival time
5 0
Enter the burst time and arrival time
7 0
Enter the burst time and arrival time
2 0
Enter the burst time and arrival time
4 0
Enter the burst time and arrival time
6 0

Sl	Bst	Arr	Turn	Wait
1	5	0	5 ms	0 ms
2	7	0	12 ms	5 ms
3	2	0	14 ms	12 ms
4	4	0	18 ms	14 ms
5	6	0	24 ms	18 ms

b) SJF

Enter the number of processes : 5

Enter your choice :
1.FCFS
2.SJF
3.RR

4.Priority 2

Enter the burst time and arrival time
2 0
Enter the burst time and arrival time
5 0
Enter the burst time and arrival time
3 0
Enter the burst time and arrival time
6 0
Enter the burst time and arrival time
1 0

Sl	Bst	Arr	Turn	Wait	
5	1	0	1 ms	0 ms	
1	2	0	3 ms	1 ms	
3	3	0	6 ms	3 ms	
2	5	0	11 ms		6 ms
4	6	0	17 ms	11 ms	

c) RR
Enter the number of processes : 3

Enter your choice :
1.FCFS
2.SJF
3.RR
4.Priority 3

Enter the burst time and arrival time
3 0
Enter the burst time and arrival time
6 0
Enter the burst time and arrival time
7 0
Enter the time slice :2

Sl	Bst	Arr	Wait	Turn
1	3	0	4 ms	7 ms
2	6	0	7 ms	13 ms
3	7	0	9 ms	16 ms

d) Priority Scheduling
Enter the number of processes : 4

Enter your choice :
1.FCFS
2.SJF
3.RR
4.Priority 4

Enter the burst time and arrival time
4 0
Enter the burst time and arrival time
2 0
Enter the burst time and arrival time
1 0
Enter the burst time and arrival time
9 0
Enter the priority: 2
3 1 1

Sl	Bst	Arr	Prio	Turn	Wait
2	4	0	3	4 ms	0 ms
1	2	0	2	6 ms	4 ms
3	1	0	1	7 ms	6 ms
4	9	0	1	16 ms	7 ms

Experiment 2:

Simulate the following file organization techniques

a) Single level directory b) Two level directory c) Hierarchical

Program :

a) Single level directory :

```
#include<stdio.h>
#include<ctype.h>
#include<stdlib.h>
#include<string.h>
struct{
   char dname[10],fname[10][10];
   int fcnt;
}dir;
void main(){

   int i,ch;
   char f[30];
   dir.fcnt = 0;
   printf("\nEnter name of directory -- ");
   scanf("%s", dir.dname);
   while(1)
   {
      printf("\n\n 1. Create File\t2. Delete File\t3. Search File \n 4. Display Files\t5.
Exit\nEnter your choice -- ");
      scanf("%d",&ch);
      switch(ch)
      {
            case 1: printf("\n Enter the name of the file -- ");
                  scanf("%s",dir.fname[dir.fcnt]);
                  dir.fcnt++;
                  break;
            case 2: printf("\n Enter 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 ",f);
                        strcpy(dir.fname[i],dir.fname[dir.fcnt-1]);
                        break;
                    }
                }
                if(i==dir.fcnt)
                    printf("File %s not found",f);
                else
                    dir.fcnt--;
                break;
            case 3: printf("\n Enter 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 ", f);
                        break;
                    }
                }
                if(i==dir.fcnt)
                    printf("File %s not found",f);
                break;
            case 4: if(dir.fcnt==0)
                        printf("\n Directory Empty");
                else
                {
                    printf("\n The Files are -- ");
                    for(i=0;i<dir.fcnt;i++)
                    printf("\t%s",dir.fname[i]);
                }
                break;
            default: exit(0);
        }
    }
}
```

b) Two level directory :

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
struct{
```

```
    char dname[10],fname[10][10];
    int fcnt;
}dir[10];
void main()
{
    int i,ch,dcnt,k;
    char f[30], d[30];
    dcnt=0;
    while(1)
    {
        printf("\n\n 1. Create Directory\t 2. Create File\t 3. Delete File");
        printf("\n 4. Search File \t \t 5. Display \t 6. Exit \t Enter your choice -- ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: printf("\n Enter name of directory -- ");
                    scanf("%s", dir[dcnt].dname);
                    dir[dcnt].fcnt=0;
                    dcnt++;
                    printf("Directory created");
                    break;
            case 2: printf("\n Enter name of the directory -- ");
                    scanf("%s",d);
                    for(i=0;i<dcnt;i++)
                        if(strcmp(d,dir[i].dname)==0)
                        {
                            printf("Enter name of the file -- ");
                            scanf("%s",dir[i].fname[dir[i].fcnt]);
                            dir[i].fcnt++;
                            printf("File created");
                            break;
                        }
                        if(i==dcnt)
                            printf("Directory %s not found",d);
                        break;
            case 3: printf("\nEnter name of the directory -- ");
                    scanf("%s",d);
                    for(i=0;i<dcnt;i++)
                    {
                        if(strcmp(d,dir[i].dname)==0)
                        {
                            printf("Enter name of the file -- ");
                            scanf("%s",f);
```

```
            for(k=0;k<dir[i].fcnt;k++)
            {
               if(strcmp(f, dir[i].fname[k])==0)
               {
                  printf("File %s is deleted ",f);
                  dir[i].fcnt--;
                  strcpy(dir[i].fname[k],dir[i].fname[dir[i].fcnt]);
                  goto jmp;
               }
            }
            printf("File %s not found",f);
            goto jmp;
         }
      }
      printf("Directory %s not found",d);
      jmp : break;
case 4: printf("\nEnter name of the directory -- ");
      scanf("%s",d);
      for(i=0;i<dcnt;i++)
      {

         if(strcmp(d,dir[i].dname)==0)
         {
            printf("Enter the name of the file -- ");
            scanf("%s",f);
            for(k=0;k<dir[i].fcnt;k++)
            {
               if(strcmp(f, dir[i].fname[k])==0)
               {
                  printf("File %s is found ",f);
                  goto jmp1;
               }
            }
            printf("File %s not found",f);
            goto jmp1;
         }
      }
      printf("Directory %s not found",d);
      jmp1: break;
case 5: if(dcnt==0)
         printf("\nNo Directory's ");
      else
      {
```

```
                    printf("\nDirectory\tFiles");
                    for(i=0;i<dcnt;i++)
                    {
                        printf("\n%s\t\t",dir[i].dname);
                        for(k=0;k<dir[i].fcnt;k++)
                            printf("\t%s",dir[i].fname[k]);
                    }
                }
                break;
            default:exit(0);
        }
    }
}
```

c) Hierarchical :

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

struct node
{
    char name[128];
    bool isDir;
    struct node *p;
    struct node *c[100];
    int i;
} * head, *curr;

void ls()
{
    if (curr->i == 0)
    {
        printf("Empty directory\n");
        return;
    }
    int i=0;
    for ( i = 0; i < curr->i; i++)
    {
        if (curr->c[i]->isDir)
            printf("*%s*  ", curr->c[i]->name);
        else
            printf("%s  ", curr->c[i]->name);
    }
```

```
}

void add(bool d)
{
    printf("Enter the Name:\n");
    char fname[128];
    scanf("%s", fname);
    struct node *temp = (struct node *)malloc(sizeof(struct node));
    strcpy(temp->name, fname);
    temp->isDir = d;
    temp->p = curr;
    temp->i=0;

    curr->c[curr->i] = temp;
    curr->i = (curr->i) + 1;
}

void cd()
{
    printf("Enter directory name:\n");
    char dname[128];
    scanf("%s", dname);
    int i=0;
    for (i = 0; i < curr->i; i++)
    {
        if (!strcmp(curr->c[i]->name, dname) && curr->c[i]->isDir)
        {
            curr = curr->c[i];
            return;
        }
    }
    printf("Directory not present.\n");
}

void cdparent()
{
    if (curr->p == NULL)
    {
        printf("You are at the root directory\n");
        return;
    }
    curr = curr->p;
}
```

```
void del(bool d)
{
   printf("Enter name of file or directory to delete:\n");
   char name[128];
   scanf("%s", name);
   int i =0;
   for ( i = 0; i < curr->i; i++)
   {
      if (!strcmp(curr->c[i]->name, name) && ((d && curr->c[i]->isDir == true) || (!d &&
curr->c[i]->isDir == false)))
      {
         int t = i;
         while (t < (curr->i) - 1)
         {
            curr->c[t] = curr->c[t + 1];
            t++;
         }
         curr->i = (curr->i) - 1;
         printf("Successfully deleted.\n");
         return;
      }
   }
   printf("Not found\n");
}

void main()
{
   int in;
   head = (struct node *)malloc(sizeof(struct node));
   strcpy(head->name, "root");
   head->isDir = true;
   head->p = NULL;
   head->i = 0;
   curr = head;
   while (true)
   {
      printf("\n\nYou are in %s directory.\n1. show everything in this directory\n2. change
directory\n3. go to parent directory\n4. add new file\n5. delete file\n6. create new
directory\n7. delete directory\n8. exit\n", curr->name);
      scanf("%d", &in);
      switch (in)
      {
```

```
                case 1:
                    ls();
                    break;
                case 2:
                    cd();
                    break;
                case 3:
                    cdparent();
                    break;
                case 4:
                    add(false);
                    break;
                case 5:
                    del(false);
                    break;
                case 6:
                    add(true);
                    break;
                case 7:
                    del(true);
                    break;
                default:
                    exit(0);
            }
        }
    }
```

Output :

a) Single level directory :
   Enter name of directory -- A

   1. Create File 2. Delete File 3. Search File
   4. Display Files 5. Exit
   Enter your choice -- 1

   Enter the name of the file -- F1

   1. Create File 2. Delete File 3. Search File

4. Display Files 5. Exit
Enter your choice -- 1

Enter the name of the file -- F2

1. Create File 2. Delete File 3. Search File
4. Display Files 5. Exit
Enter your choice -- 4

The Files are -- F1 F2

1. Create File 2. Delete File 3. Search File
4. Display Files 5. Exit
Enter your choice -- 3

Enter the name of the file -- F1
File F1 is found

1. Create File 2. Delete File 3. Search File
4. Display Files 5. Exit
Enter your choice -- 2

Enter the name of the file -- F2
File F2 is deleted

1. Create File 2. Delete File 3. Search File
4. Display Files 5. Exit
Enter your choice -- 4

The Files are -- F1

1. Create File 2. Delete File 3. Search File
4. Display Files 5. Exit
Enter your choice -- 5

b) Two level directory :
1. Create Directory 2. Create File 3. Delete File
4. Search File 5. Display 6. Exit Enter your choice -- 1

Enter name of directory -- D1
Directory created

```
 1. Create Directory     2. Create File  3. Delete File
 4. Search File                  5. Display       6. Exit          Enter your choice -- 1

 Enter name of directory -- D2
Directory created

 1. Create Directory     2. Create File  3. Delete File
 4. Search File                  5. Display       6. Exit          Enter your choice -- 5

Directory        Files
D1
D2

 1. Create Directory     2. Create File  3. Delete File
 4. Search File                  5. Display       6. Exit          Enter your choice -- 2

Enter name of the directory -- D1
Enter name of the file -- F1
File created

 1. Create Directory     2. Create File  3. Delete File
 4. Search File                  5. Display       6. Exit          Enter your choice -- 5

Directory        Files
D1                       F1
D2

 1. Create Directory     2. Create File  3. Delete File
 4. Search File                  5. Display       6. Exit          Enter your choice -- 4

Enter name of the directory -- D1
Enter the name of the file -- F1
File F1 is found

 1. Create Directory     2. Create File  3. Delete File
 4. Search File                  5. Display       6. Exit          Enter your choice -- 3

Enter name of the directory -- D1
Enter name of the file -- F1
File F1 is deleted

 1. Create Directory     2. Create File  3. Delete File
 4. Search File                  5. Display       6. Exit          Enter your choice -- 4
```

1. Create Directory 2. Create File 3. Delete File
4. Search File 5. Display 6. Exit Enter your choice -- 5

Directory Files
D1
D2

1. Create Directory 2. Create File 3. Delete File
4. Search File 5. Display 6. Exit Enter your choice -- 6

c) Hierarchical

You are in root directory.
1. show everything in this directory 2. change directory 3. go to parent directory
4. add new file 5. delete file 6. create new directory 7. delete directory 8. exit
6
Enter the Name:
D1

You are in root directory.
1. show everything in this directory 2. change directory 3. go to parent directory
4. add new file 5. delete file 6. create new directory 7. delete directory 8. exit
1
D1

You are in root directory.
1. show everything in this directory 2. change directory 3. go to parent directory
4. add new file 5. delete file 6. create new directory 7. delete directory 8. exit
2
Enter directory name:
D1

You are in D1 directory.
1. show everything in this directory 2. change directory 3. go to parent directory
4. add new file 5. delete file 6. create new directory 7. delete directory 8. exit
4
Enter the Name:
F1

You are in D1 directory.
1. show everything in this directory 2. change directory 3. go to parent directory
4. add new file 5. delete file 6. create new directory 7. delete directory 8. exit
5
Enter name of file or directory to delete:
F1
Successfully deleted.

You are in D1 directory.
1. show everything in this directory 2. change directory 3. go to parent directory
4. add new file 5. delete file 6. create new directory 7. delete directory 8. exit
3

You are in root directory.
1. show everything in this directory 2. change directory 3. go to parent directory
4. add new file 5. delete file 6. create new directory 7. delete directory 8. exit
1
D1

You are in root directory.
1. show everything in this directory 2. change directory 3. go to parent directory
4. add new file 5. delete file 6. create new directory 7. delete directory 8. exit
7
Enter name of file or directory to delete:
D1
Successfully deleted.

You are in root directory.
1. show everything in this directory 2. change directory 3. go to parent directory
4. add new file 5. delete file 6. create new directory 7. delete directory 8. exit
1
Empty directory

You are in root directory.
1. show everything in this directory 2. change directory 3. go to parent directory
4. add new file 5. delete file 6. create new directory 7. delete directory 8. exit
8

Experiment 3 :

Implement the banker's algorithm for deadlock avoidance.

Program :

```
#include<stdio.h>
#include<stdlib.h>
void main(){
   int p=0,r=0,i,j,count=0,flag;


   //Entering the number of processes and resources
   printf("Enter the number of processes and resources : ");
   scanf("%d %d",&p,&r);
   printf("\n");

////////////////////////////////////////////////////////////

   //Needed data structures
   int alloc[p][r],num[r],available[r],max[p][r],need[p][r],work[r],finish[p],order[p];

////////////////////////////////////////////////////////////

   //Allocation matrix input
   printf("Enter the resource allocation table : \n");
   printf("--------------------------------------\n");
   for(i=0;i<p;i++){
      finish[i]=0;
      for(j=0;j<r;j++){
         scanf("%d",&alloc[i][j]);
      }
   }
   /*
   //Output for alloc
   for(int i=0;i<p;i++){
      for (int j = 0; j < r; j++)
      {
         printf("%d  ",alloc[i][j]);
      }
```

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

////////////////////////////////////////////////////////

    //Maximum Number available
    for ( i = 0; i < r; i++)
    {
        printf("Enter the number of resources available of Type %d : ",i+1);
        scanf("%d",&num[i]);
    }
    /*
    //output for num
    for (int i = 0; i < r; i++)
    {
        printf("%d  ",num[i]);
    }
    */

////////////////////////////////////////////////////////
    //Counting each resource allocated
    for ( i = 0; i < r; i++)
    {   available[i]=num[i];
        work[i]=num[i];
        for ( j = 0; j < p; j++)
        {
            available[i]-=alloc[j][i];
            work[i]-=alloc[j][i];
        }
    }
    /*
    for (int i = 0; i < r; i++)
    {
        printf("%d  ",available[i]);
    }
    */

////////////////////////////////////////////////////////
    //Maximum Matrix
    printf("Enter the Maximum resource table : \n");
    printf("--------------------------------------\n");
    for(i=0;i<p;i++){
```

```
        for(j=0;j<r;j++){
            scanf("%d",&max[i][j]);
        }
    }

////////////////////////////////////////////////////////

    //Need Matrix
    for ( i = 0; i < p; i++)
    {
        for ( j = 0; j < r; j++)
        {
            need[i][j]=max[i][j]-alloc[i][j];
        }

    }
    //Output for Need
    for(int i=0;i<p;i++){
        for (int j = 0; j < r; j++)
        {
            printf("%d  ",need[i][j]);
        }
        printf("\n");
    }

////////////////////////////////////////////////////////

//SAFTEY ALGORITHM

while (count<p)
{
    flag=0;
    for ( i = 0; i < p; i++)
    {
        if (finish[i] == 0)
        {
            for (j = 0; j < r; j++)
                if (need[i][j]>work[j])
                    break;
            if (j == r)
            {
                for ( int k = 0; k < r; k++)
                        work[k] += alloc[i][k];
```

```
            order[count]=i;
            count++;
            finish[i]=1;
            flag=1;
          }


      }
   }
  if (flag == 0)
   {
      printf("System is not in safe state");
      exit(0);
   }
}
      printf("System is in safe state\n Order is : \t");
      for(i=0;i<p;i++)
         printf(" P%d -",order[i]+1);
      printf("\n");
}
```

Output :

```
Enter the number of processes and resources : 5 3
Enter the resource allocation table :
--------------------------------------
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter the number of resources available of Type 1 : 10
Enter the number of resources available of Type 2 : 5
Enter the number of resources available of Type 3 : 7
Enter the Maximum resource table :
--------------------------------------
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
7  4  3
1  2  2
```

```
6  0  0
0  1  1
4  3  1
System is in safe state
 Order is :        P2 - P4 - P5 - P1 - P3 -
```

Experiment 4:

Implement the producer-consumer problem using semaphores.

Program :

```
#include<stdio.h>
#include<ctype.h>
#include<stdlib.h>
int empty=0,full=0,n=0,mutex=1;
int pro_cnt=0,con_cnt=0;
int wait(int);
int signal(int);
void main(){
  int choice;
  void producer();
  void consumer();
  printf("Enter the buffer size : ");
  scanf("%d",&n);
  empty=n;
  printf("Enter your choice :-\n");
  printf("1.Produce\t2.Consume\t3.Exit\n");
  while(1){
    printf("Your choice : ");
    scanf("%d",&choice);
    printf("\n");
    switch(choice){
      case 1:{  if((mutex==1)&&(empty!=0))
                  producer();
                else
                  printf("Buffer is full!!!!\n");
                break;
               }
      case 2:{  if((mutex==1)&&(full!=0))
                  consumer();
                else
                  printf("Buffer is empty!!!!!\n");
                break;
```

```
            }
      default:exit(0);break;
      }
    printf("Number of produced items : %d \nNumber of consumed items : %d
\n",pro_cnt,con_cnt);
  }
}
//Producer Function
void producer(){
    mutex=wait(mutex);
    full=signal(full);
    empty=wait(empty);
    pro_cnt++;
    printf("Produced Item.. \n");
    mutex=signal(mutex);
}
//Consumer Function
void consumer(){
  mutex=wait(mutex);
  full=wait(full);
  empty=signal(empty);
  con_cnt++;
  printf("Consumed Item.. \n");
  mutex=signal(mutex);
}
//Wait
int wait(int semaphore){
      return(--semaphore);
}
//Signal
int signal(int semaphore){
      return(++semaphore);
}
```

Output :

```
          Enter the buffer size : 3
          Enter your choice :-
          1.Produce       2.Consume      3.Exit
          Your choice : 1

          Produced Item..
          Number of produced items : 1
```

```
Number of consumed items : 0
Your choice : 1

Produced Item..
Number of produced items : 2
Number of consumed items : 0
Your choice : 2

Consumed Item..
Number of produced items : 2
Number of consumed items : 1
Your choice : 1

Produced Item..
Number of produced items : 3
Number of consumed items : 1
Your choice : 1

Produced Item..
Number of produced items : 4
Number of consumed items : 1
Your choice : 1

Buffer is full!!!!
Number of produced items : 4
Number of consumed items : 1
Your choice : 2

Consumed Item..
Number of produced items : 4
Number of consumed items : 2
Your choice : 2

Consumed Item..
Number of produced items : 4
Number of consumed items : 3
Your choice : 2

Consumed Item..
Number of produced items : 4
Number of consumed items : 4
Your choice : 2
```

Buffer is empty!!!!!
Number of produced items : 4
Number of consumed items : 4
Your choice : 3

Experiment 5 :

Write a program to simulate the working of the dining philosopher's problem.

Program :

```
#include<stdio.h>
#include<stdlib.h>
void pickup(int);
void putdown(int);
void test(int);
void initialize();
enum{THINKING,HUNGRY,EATING} state[5];
int flag[5];
void main(){
  int phil=0,choice=0;
  initialize();
  while(1){
    printf("Enter the philosopher number(1-5) : ");
    scanf("%d",&phil);
    printf("Enter the operation needed \n 1.Start Eating\t2.Stop Eating\t 3.Exit\n");
    scanf("%d",&choice);
    switch(choice){
       case 1:{
            pickup(phil);
            break;           }
       case 2:{
            putdown(phil);
            break;          }
       default:exit(0);break;
    }     }         }
void initialize(){
  int i=0;
  for(i=0;i<5;i++){
    state[i]=THINKING;
    flag[i]=0;      }         }
void pickup(int id){
  state[id]=HUNGRY;
  test(id);
```

```
  if(state[id]!=EATING)
     flag[id]=0;

   if(flag[id]==1)
     printf("Philosopher %d is eating!\n",id);
   else
     printf("Philosopher %d cannot eat\n",id);
}
void putdown(int id){
  state[id]=THINKING;
  test((id+4)%5);
  test((id+1)%5);
  printf("Philosopher %d is thinking!\n",id);     }
void test(int i){
  if((state[(i+4)%5]!=EATING) && (state[i]==HUNGRY) && (state[(i+1)%5]!=EATING)){
  state[i]=EATING;
  flag[i]=1;        }
}
```

Output :

```
        Enter the philosopher number(1-5) : 1
        Enter the operation needed
         1.Start Eating 2.Stop Eating   3.Exit
        1
        Philosopher 1 is eating!
        Enter the philosopher number(1-5) : 4
        Enter the operation needed
         1.Start Eating 2.Stop Eating   3.Exit
        1
        Philosopher 4 is eating!
        Enter the philosopher number(1-5) : 3
        Enter the operation needed
         1.Start Eating 2.Stop Eating   3.Exit
        1
        Philosopher 3 cannot eat
        Enter the philosopher number(1-5) : 1
        Enter the operation needed
         1.Start Eating 2.Stop Eating   3.Exit
        2
        Philosopher 1 is thinking!
        Enter the philosopher number(1-5) : 2
        Enter the operation needed
         1.Start Eating 2.Stop Eating   3.Exit
```

```
        1
        Philosopher 2 is eating!
        Enter the philosopher number(1-5) : 3
        Enter the operation needed
         1.Start Eating 2.Stop Eating   3.Exit
```

Experiment 6 :

Simulate the following disk scheduling algorithms.
a) FCFS b)SCAN c) C-SCAN

Program :

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
void main(){
  int choice,n,start,seek_time,range,temp,index=0,count=0;
  printf("DISK SCHEDULING \n");
  printf("--------------------\n");
  printf("Enter the range : ");
  scanf("%d",&range);
  printf("Enter the number of disk requests : ",range);
  scanf("%d",&n);
  printf("Enter the head position (< %d): ",range);
  scanf("%d",&start);
  if(start>range)
     exit(0);
  printf("Enter the requests : ");
  int disk_req[n],order[n],a[n];
  disk_req[0]=start;
  a[0]=start;
  for(int i=1;i<=n;i++){
     scanf("%d",&disk_req[i]);
     a[i]=disk_req[i];
  }
  for(int i=0;i<=n;i++){
    for(int j=0;j<=n-i-1;j++){
          if(a[j]>a[j+1]){
            temp=a[j];
            a[j]=a[j+1];
            a[j+1]=temp;
          }
    }
  }
  while(1){
```

```
seek_time=0;
printf("Enter your choice \n1.FCFS\t2.SCAN\t3.C-SCAN\n");
scanf("%d",&choice);
switch(choice){
  case 1:{  printf("No\tDisk Req\n");
            order[0]=1;
            printf("%d\t%d\n",order[0],disk_req[0]);
            for(int i=1;i<=n;i++){
               order[i]=i+1;
               printf("%d\t%d\n",order[i],disk_req[i]);
               seek_time+=abs(disk_req[i]-disk_req[i-1]);
           }
           printf("Seek Time : %d ms \n",seek_time);
           break;
  }
  case 2:{
            printf("No\tDisk Req\n");
            for(int i=0;i<=n;i++){
                if(a[i]==start){
                    index=i;
                 }
            }
            for(int i=index-1;i>=0;i--){
               printf("%d\t%d\n",count,a[i]);
               seek_time+=abs(a[i+1]-a[i]);
               //printf("Seek time :%d\n",seek_time);
               count++;
               if(i==0){
                 printf("%d\t%d\n",count,0);
                 seek_time+=a[0];
                 count++;
               }
            }
            for(int i=index+1;i<n;i++){
               if(i==(index+1)){
                  seek_time+=a[i];
                 //  printf("Seek time :%d\n",seek_time);
               }
               printf("%d\t%d\n",count,a[i]);
               seek_time+=abs(a[i+1]-a[i]);
               count++;
               //printf("Seek time :%d\n",seek_time);
               if(i==(n-1))
```

```
                printf("%d\t%d\n",count,a[n]);
            }
            printf("Seek Time : %d ms\n",seek_time);
           break;
    }
    case 3:{  count=0;
            printf("No\tDisk Req\n");
            for(int i=0;i<=n;i++){
                if(a[i]==start){
                    index=i;
                }
            }
            for(int i=index-1;i>=0;i--){
                printf("%d\t%d\n",count,a[i]);
                seek_time+=abs(a[i+1]-a[i]);
                //printf("Seek time :%d\n",seek_time);
                count++;
                if(i==0){
                    printf("%d\t%d\n",++count,0);
                    seek_time+=(a[0]+range);
                    printf("%d\t%d\n",++count,range-1);
                    count++;
                }
            }
            for(int i=(n-1);i>index;i--){
                printf("%d\t%d\n",count,a[i]);
                seek_time+=abs(a[i+1]-a[i]);
                count++;
            }
            printf("Seek Time : %d ms \n",seek_time);
           break;
    }
    default:exit(0);break;
   }
  }
}
```

Output :

```
        DISK SCHEDULING
        --------------------
        Enter the range : 200
        Enter the number of disk requests : 5
        Enter the head position (< 200): 40
```

```
Enter the requests : 10 20 30 50 60

Enter your choice
1.FCFS          2.SCAN          3.C-SCAN        4.Exit
1
No      Disk Req
1       40
2       10
3       20
4       30
5       50
6       60
Seek Time : 80 ms

Enter your choice
1.FCFS          2.SCAN          3.C-SCAN        4.Exit
2
No      Disk Req
0       30
1       20
2       10
3       0
4       50
5       60
Seek Time : 100 ms

Enter your choice
1.FCFS          2.SCAN          3.C-SCAN        4.Exit
3
No      Disk Req
0       30
1       20
2       10
4       0
5       199
6       50
Seek Time : 250 ms

Enter your choice
1.FCFS          2.SCAN          3.C-SCAN        4.Exit
4
```

Experiment 7 :

Implement pass one of a two pass assembler.

Program :

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
void main()
{
  FILE *f1,*f2,*f3,*f4,*f5;
  int lc,sa,l,op1,o,len;
  char m1[20],la[20],op[20],otp[20];
  f1=fopen("input.txt","r");
  f3=fopen("symtab.txt","w");
  f4=fopen("length.txt","w");
  f5=fopen("start.txt","w");
  fscanf(f1,"%s %s %d",la,m1,&op1);
  if(strcmp(m1,"START")==0)
  {
    sa=op1;
    fprintf(f5,"%d",sa);
    lc=sa;
    printf("\t%s\t%s\t%d\n",la,m1,op1);
  }
  else
    lc=0;
  fscanf(f1,"%s %s",la,m1);
  while(!feof(f1))
   {
      fscanf(f1,"%s",op);
      printf("\n%d\t%s\t%s\t%s\n",lc,la,m1,op);
      if(strcmp(la,"-")!=0)
      {
          fprintf(f3,"\n%d\t%s\n",lc,la);
      }
      f2=fopen("optab.txt","r");
      fscanf(f2,"%s %d",otp,&o);
      while(!feof(f2))
```

```
 {
    if(strcmp(m1,otp)==0)
    {
         lc=lc+3;
         break;
    }
  fscanf(f2,"%s %d",otp,&o);
 }
 fclose(f2);
 if(strcmp(m1,"WORD")==0)
 {
   lc=lc+3;
 }
 else if(strcmp(m1,"RESW")==0)
  {
    op1=atoi(op);
    lc=lc+(3*op1);
  }
 else if(strcmp(m1,"BYTE")==0)
 {
    if(op[0]=='X')
      lc=lc+1;
    else
    {
      len=strlen(op)-2;
      lc=lc+len;
    }
 }
 else if(strcmp(m1,"RESB")==0)
 {
  op1=atoi(op);
  lc=lc+op1;
 }
 fscanf(f1,"%s%s",la,m1);
 }
if(strcmp(m1,"END")==0)
{
printf("Program length = %d \n",lc-sa);
fprintf(f4,"%d",lc-sa);
}
fclose(f1);
fclose(f3);
fclose(f4);
```

}

Output :

Input.txt :

```
COPY    START   1000
-       LDA     ALPHA
-       ADD     ONE
-       SUB     TWO
-       STA     BETA
ALPHA   BYTE    C'KLNCE
ONE     RESB    2
TWO     WORD    5
BETA    RESW    1
-       END     -
```

Console :

```
                COPY    START           1000

        1000    -       LDA     ALPHA
        1003    -       ADD     ONE
        1006    -       SUB     TWO
        1009    -       STA     BETA
        1012    ALPHA   BYTE    C'KLNCE
        1017    ONE     RESB    2
        1019    TWO     WORD    5
        1022    BETA    RESW    1
        1025    -       END     -
        Program length = 25
```

Experiment 8:

Implement pass two of a two pass assembler.

Program :

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

void main()
   {
   FILE *fint,*ftab,*flen,*fsym;
   int op1[10],txtlen,txtlen1,i,j=0,len;
   char
add[5],symadd[5],op[5],start[10],temp[30],line[20],label[20],mne[10],operand[10],symtab[10],op
mne[10];

   fint=fopen("input2.txt","r");
   flen=fopen("length.txt","r");
   ftab=fopen("optab.txt","r");
   fsym=fopen("symtab.txt","r");
   fscanf(fint,"%s%s%s%s",add,label,mne,operand);

   if(strcmp(mne,"START")==0)
   {
            strcpy(start,operand);
            fscanf(flen,"%d",&len);
   }
   printf("H^%s^%s^%d\nT^00%s^",label,start,len,start);

         fscanf(fint,"%s%s%s%s",add,label,mne,operand);
   while(strcmp(mne,"END")!=0)
   {
            fscanf(ftab,"%s%s",opmne,op);
            while(!feof(ftab))
                  {
                           if(strcmp(mne,opmne)==0)
                           {
```

```
                        fclose(ftab);
                        fscanf(fsym,"%s%s",symadd,symtab);
                        while(!feof(fsym))
                        {
                                if(strcmp(operand,symtab)==0)
                                {
                                        printf("%s%s^",op,symadd);
                                        break;
                                }
                                else
                                        fscanf(fsym,"%s%s",symadd,symtab);
                        }
                                break;
                }
                else
                        fscanf(ftab,"%s%s",opmne,op);
        }
        if((strcmp(mne,"BYTE")==0)||(strcmp(mne,"WORD")==0))
        {
                if(strcmp(mne,"WORD")==0)
                         printf("0000%s^",operand);
                else
                {
                        len=strlen(operand);
                        for(i=2;i<len;i++)
                        {
                                printf("%d",operand[i]);
                        }
                        printf("^");
                }
        }
        fscanf(fint,"%s%s%s%s",add,label,mne,operand);
        ftab=fopen("optab.txt","r");
        fseek(ftab,SEEK_SET,0);
}

printf("\nE^00%s",start);
fclose(fint);
fclose(ftab);
fclose(fsym);
fclose(flen);
}
```

Output :

Intermediate.txt :

	COPY	START	1000
1000	-	LDA	ALPHA
1003	-	ADD	ONE
1006	-	SUB	TWO
1009	-	STA	BETA
1012	ALPHA	BYTE	C'KLNCE
1017	ONE	RESB	2
1019	TWO	WORD	5
1022	BETA	RESW	1
1025	-	END	-

Length.txt :

25

Console :

H^COPY^1000^25
T^001000^001012^011017^051019^231022^7576786769^00005^
E^001000

Experiment 9:

Implement a single pass assembler.

Program :

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
void main()
{
     FILE *f1,*f2,*f3,*f4,*f5;
     int lc,sa,i=0,j=0,m[10],pgmlen,len,k,len1,l=0;
     char name[10],opnd[10],la[10],mne[10],s1[10],mne1[10],opnd1[10];
     char lcs[10],ms[10];
     char sym[10],symaddr[10],obj1[10],obj2[10],s2[10],q[10],s3[10];
     f1=fopen("input3.txt","r");
     f2=fopen("optab3.txt","r");
     f3=fopen("symtab3.txt","w+");
     f4=fopen("symtab13.txt","w+");
     f5=fopen("output3.txt","w+");
     fscanf(f1,"%s%s%s",la,mne,opnd);
     if(strcmp(mne,"START")==0)
    {
       sa=atoi(opnd);
       strcpy(name,la);
       lc=sa;
    }
    strcpy(s1,"*");
    fscanf(f1,"%s%s%s",la,mne,opnd);
    while(strcmp(mne,"END")!=0)
    {
      if(strcmp(la,"-")==0)
      {
        fscanf(f2,"%s%s",mne1,opnd1);
        while(!feof(f2))
        {
          if(strcmp(mne1,mne)==0)
          {
       m[i]=lc+1;
```

```
 fprintf(f3,"%s\t%s\n",opnd,s1);
 fprintf(f5,"%s\t0000\n",opnd1);
 lc=lc+3;
 i=i+1;
 break;
    }
    else
  fscanf(f2,"%s%s",mne1,opnd1);
  }
}
else
{
  fseek(f3,SEEK_SET,0);
  fscanf(f3,"%s%s",sym,symaddr);
  while(!feof(f3))
  {
    if(strcmp(sym,la)==0)
    {
 //itoa(lc,lcs,10);
     snprintf(lcs,10,"%d",lc);
 fprintf(f4,"%s\t%s\n",la,lcs);
     //itoa(m[j],ms,10);
     snprintf(ms,10,"%d",m[j]);
 j=j+1;
 fprintf(f5,"%s\t%s\n",ms,lcs);
 i=i+1;
 break;
    }
    else
  fscanf(f3,"%s%s",sym,symaddr);
  }  //f3
  if(strcmp(mne,"RESW")==0)
   lc=lc+3*atoi(opnd);
  else if(strcmp(mne,"BYTE")==0)
  {
   strcpy(s2,"-");
   len=strlen(opnd);
   lc=lc+len-2;
   for(k=2;k<len;k++)
   {
   q[l]=opnd[k];
   l=l+1;
   }
```

```
    fprintf(f5,"%s\t%s\n",q,s2);
    break;
   }
   else if(strcmp(mne,"RESB")==0)
    lc=lc+atoi(opnd);
   else if(strcmp(mne,"WORD")==0)
   {
     strcpy(s3,"#");
     lc=lc+3;
     fprintf(f5,"%s\t%s\n",opnd,s3);
     break;
   }
  } // else la=-
   fseek(f2,SEEK_SET,0);
   fscanf(f1,"%s%s%s",la,mne,opnd);
}
fseek(f5,SEEK_SET,0);
pgmlen=lc-sa;
printf("H^%s^%d^0%x\n",name,sa,pgmlen);
printf("T^");
printf("00%d^0%x",sa,pgmlen);
fscanf(f5,"%s%s",obj1,obj2);
while(!feof(f5))
{
  if(strcmp(obj2,"0000")==0)
    printf("^%s%s",obj1,obj2);
  else if(strcmp(obj2,"-")==0)
    {
    printf("^");
    len1=strlen(obj1);
    for(k=0;k<len1;k++)
    printf("%d",obj1[k]);
    }
    else if(strcmp(obj2,"#")==0)
    {
     printf("^");
     printf("%s",obj1);
     }
  fscanf(f5,"%s%s",obj1,obj2);
}
fseek(f5,SEEK_SET,0);
fscanf(f5,"%s%s",obj1,obj2);
while(!feof(f5))
```

```
{
 if(strcmp(obj2,"0000")!=0)
 {
  if(strcmp(obj2,"-")!=0)
  {
  if(strcmp(obj2,"#")!=0)
  {
   printf("\n");
   printf("T^%s^02^%s",obj1,obj2);
  }
 }
 }
  fscanf(f5,"%s%s",obj1,obj2);
 }
 printf("\nE^00%d",sa);
}
```

Output :

Input.txt :

```
COPY        START           1000
-           LDA         ALPHA
-           STA         BETA
ALPHA       RESW        1
BETA        RESW        1
-           END         -
```

Console :

```
H^COPY^1000^0c
T^001000^0c^000000^230000
T^1001^02^1006
T^1004^02^1009
E^001000%
```

Experiment 10 :

Implement a two pass macro processor.

Program :

Pass 1 :

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

void main()
{
   FILE *f1,*f2,*f3;
   char mne[20],opnd[20],la[20];
   f1=fopen("input.txt","r");
   f2=fopen("namtab.txt","w+");
   f3=fopen("deftab.txt","w+");
   fscanf(f1,"%s%s%s",la,mne,opnd);
   while(strcmp(mne,"MEND")!=0)
   {
      if(strcmp(mne,"MACRO")==0)
      {
         fprintf(f2,"%s\n",la);
         fprintf(f3,"%s\t%s\n",la,opnd);
      }
      else
         fprintf(f3,"%s\t%s\n",mne,opnd);
      fscanf(f1,"%s%s%s",la,mne,opnd);
   }
   fprintf(f3,"%s",mne);
   fclose(f1);
   fclose(f2);
   fclose(f3);
   printf("\nPass 1 of 2 pass macroprocessor is successful.");
}
```

Pass 2 :

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

```
#include<stdlib.h>

void main()
{
   FILE *f1,*f2,*f3,*f4,*f5;
   int i,len;
   char mne[20],opnd[20],la[20],name[20],mne1[20],opnd1[20],arg[20];
   f1=fopen("input.txt","r");
   f2=fopen("namtab.txt","r");
   f3=fopen("deftab.txt","r");
   f4=fopen("argtab.txt","w+");
   f5=fopen("output.txt","w");
   fscanf(f1,"%s%s%s",la,mne,opnd);
   while(strcmp(mne,"END")!=0)
   {
      if(strcmp(mne,"MACRO")==0)
      {
         fscanf(f1,"%s%s%s",la,mne,opnd);
         while(strcmp(mne,"MEND")!=0)
            fscanf(f1,"%s%s%s",la,mne,opnd);
      }
      else
      {
         fscanf(f2,"%s",name);
         if(strcmp(mne,name)==0)
         {
            len=strlen(opnd);
            for(i=0;i<len;i++)
            {
               if(opnd[i]!=',')
                  fprintf(f4,"%c",opnd[i]);
               else
                  fprintf(f4,"\n");
            }
            fseek(f2,SEEK_SET,0);
            fseek(f4,SEEK_SET,0);
            fscanf(f3,"%s%s",mne1,opnd1);
            fprintf(f5,".\t%s\t%s\n",mne1,opnd);
            fscanf(f3,"%s%s",mne1,opnd1);
            while(strcmp(mne1,"MEND")!=0)
            {
               if((opnd1[0]=='&'))
               {
```

```
                fscanf(f4,"%s",arg);
                fprintf(f5,"-\t%s\t%s\n",mne1,arg);
              }
              else
                fprintf(f5,"-\t%s\t%s\n",mne1,opnd1);
              fscanf(f3,"%s%s",mne1,opnd1);
            }
          }
          else
            fprintf(f5,"%s\t%s\t%s\n",la,mne,opnd);
        }
        fscanf(f1,"%s%s%s",la,mne,opnd);
    }
    fprintf(f5,"%s\t%s\t%s\n",la,mne,opnd);
    fclose(f1);
    fclose(f2);
    fclose(f3);
    fclose(f4);
    fclose(f5);
    printf("\nPass 2 of 2 Pass Macroprocessor is Successful.");
}
```

Output :

Pass 1 :

Input.txt :

```
EX1     MACRO       &A,&B
-       LDA    &A
-       STA    &B
-       MEND   -
SAMPLE      START         1000
-       EX1     N1,N2
N1      RESW    1
N2      RESW    1
-       END    -
```

Deftab.txt :

```
EX1    &A,&B
LDA    &A
STA    &B
MEND
```

Namtab.txt :

```
EX1
```

Pass 2 :

Input.txt :

```
EX1     MACRO       &A,&B
-       LDA    &A
-       STA    &B
-       MEND   -
SAMPLE      START       1000
-       EX1     N1,N2
N1      RESW    1
N2      RESW    1
-       END     -
```

Output.txt :

```
SAMPLE      START               1000
.           EX1         N1,N2
-           LDA         N1
-           STA         N2
N1          RESW        1
N2          RESW        1
-           END         -
```

Experiment 11:

Implement an absolute loader.

Program :

```
#include<stdio.h>
#include<string.h>
char input[10],label[10],ch1,ch2;
int addr, w=0, start, ptaddr, l, length=0, end, count=0, k, taddr, address, i=0;
FILE *fp1,*fp2;
void check();
void main() {
  fp1=fopen("INPUT.dat","r");
  fp2=fopen("OUTPUT.dat","w");
  fscanf(fp1,"%s",input);
  printf("\n\n\t\t\t\tABSOLUTE LOADER\n");
  fprintf(fp2,"\n-----------------------------------------------------\n");
  fprintf(fp2,"MEMORY ADDRESS\t\t\tCONTENTS");
  fprintf(fp2,"\n-----------------------------------------------------\n");
  while(strcmp(input,"E")!=0)  {
   if(strcmp(input,"H")==0)  {
    fscanf(fp1,"%s %x %x %s",label,&start,&end,input);
    address=start;
   }
   else if(strcmp(input,"T")==0) {
    l=length;
    ptaddr=addr;
    fscanf(fp1,"%x %x %s",&taddr,&length,input);
    addr=taddr;
    if(w==0) {
     ptaddr=address;
     w=1;
    }
    for(k=0;k<(taddr-(ptaddr+l));k++) {
     address=address+1;
     fprintf(fp2,"xx");
     count++;
     if(count==4) {
      fprintf(fp2,"  ");
```

```
      i++;
      if(i==4) {
       fprintf(fp2,"\n\n%x\t\t",address);
       i=0;
      }
      count=0;
     }     }
    if(taddr==start)
     fprintf(fp2,"\n\n%x\t\t",taddr);
    fprintf(fp2,"%c%c",input[0],input[1]);
    check();
    fprintf(fp2,"%c%c",input[2],input[3]);
    check();
    fprintf(fp2,"%c%c",input[4],input[5]);
    check();
    fscanf(fp1,"%s",input);
   }
   else {
    fprintf(fp2,"%c%c",input[0],input[1]);
    check();
    fprintf(fp2,"%c%c",input[2],input[3]);
    check();
    fprintf(fp2,"%c%c",input[4],input[5]);
    check();
    fscanf(fp1,"%s",input);
   }       }
  fprintf(fp2,"\n-------------------------------------------------------\n");
  fclose(fp1);
  fclose(fp2);
  printf("\n\n The contents of output file:\n\n");
  fp2=fopen("OUTPUT.dat","r");
  ch2=fgetc(fp2);
  while(ch2!=EOF) {
   printf("%c",ch2);
   ch2=fgetc(fp2);
  }
  fclose(fp1);
  fclose(fp2);
}
void check() {
  count++;
  address++;
  taddr=taddr+1;
```

```
 if(count==4) {
  fprintf(fp2,"  ");
  i++;
  if(i==4) {
   fprintf(fp2,"\n\n%x\t\t",taddr);
   i=0;
  }
  count=0;
 }       }
```

Output :

Input.dat :

```
H COPY 001000 00107A
T 001000 1E 141033 482039 001036 281030 301015 482061 3C1003 00102A
0C1039 00102D
T 00101E 15 0C1036 482061 081033 4C0000 454F46 000003 000000
T 001047 1E 041030 001030 E0205D 30203F D8205D 281030 302057 549039
2C205E 38203F
T 001077 1C 101036 4C0000 000000 001000 041030 E02079 302064 509039
DC2079 2C1036
E 001000
```

Output.dat/Console :

MEMORY ADDRESS	CONTENTS			
1000	14103348	20390010	36281030	30101548
1010	20613C10	0300102A	0C103900	102D0C10
1020	36482061	0810334C	0000454F	46000003
1030	000000xx	xxxxxxxx	xxxxxxxx	xxxxxxxx
1040	xxxxxxxx	xxxxxx04	10300010	30E0205D
1050	30203FD8	205D2810	30302057	5490392C
1060	205E3820	3Fxxxxxx	xxxxxxxx	xxxxxxxx
1070	xxxxxxxx	xxxxxx10	10364C00	00000000
1080	00100004	1030E020	79302064	509039DC
1090	20792C10	36		

Experiment 12 :

Implement a symbol table with suitable hashing.

Program :

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
#define MAX 11
char l[10];
 struct symb {
    int add;
    char label[10];
}sy[11];
void search() {
    FILE *fp1;
    char la[10];
    int set=0,s;
    int j,i;
    printf("\nEnter the Label:");
    scanf("%s",la);
    fp1=fopen("symbol.txt","r");
    for(i=0;i<MAX;i++) {
        fscanf(fp1,"%d%d",&j,&sy[i].add);
        if(sy[i].add!=0)
            fscanf(fp1,"%s",sy[i].label);
    }
    for(i=0;i<MAX;i++) {
        if(sy[i].add!=0) {
            if(strcmp(sy[i].label,la)==0) {
                set=1;
                s=sy[i].add;
            }
        }
    }
    if(set==1)
        printf("\nThe Label --%s-- is present in the Symbol Table at Address:%d",la,s);
    else
```

```
        printf("\nThe Label is Not Present in the Symbol Table!!");
}
void display(int a[MAX]) {
    FILE *fp;
    int i;
    fp=fopen("symbol.txt","w");
    printf("\nThe Symbol Table");
    printf("\n*****************");
    printf("\nHash Values\tAddress\tLabel");
    for(i=0;i<MAX;i++) {
        printf("\n%d\t %d\t %s",i,sy[i].add,sy[i].label);
        fprintf(fp,"\n%d %d %s",i,sy[i].add,sy[i].label);
    }
    fclose(fp);
}
int create(int num) {
    int key;
    key=num%11;
    return key;
}
void lprob(int a[MAX],int key,int num) {
    int flag,i,count=0;
    flag=0;
    if(a[key]==0) {
        a[key]=num;
        sy[key].add=num;
        strcpy(sy[key].label,l);
    }
    else {
        i=0;
        while(i<MAX) {
            if(a[i]!=0)
                count++;
            i++;
        }
        if(count==MAX) {
            printf("\nHash table is Full!!");
            display(a);
            exit(1);
        }
        for(i=key+1;i<MAX;i++)
            if(a[i]==0) {
                a[i]=num;
```

```
            flag=1;
            sy[key].add=num;
            strcpy(sy[key].label,l);
            break;
         }
      for(i=0;i<key && flag==0;i++)
         if(a[i]==0) {
            a[i]=num;
            flag=1;
            sy[key].add=num;
            strcpy(sy[key].label,l);
            break;
         }
     }
}
void main() {
   int a[MAX],num,key,i,ch;
   char ans='y';
   for(i=0;i<MAX;i++)
      a[i]=0;
   do {
      printf("\nSymbol Table Menu\n1.Create a Symbol Table\n2.Search in the Symbol
Table\n3.Exit\nEnter your choice:");
      scanf("%d",&ch);
      switch(ch) {
         case 1: while(ans=='y') {
                  printf("\nEnter the Address:");
                  scanf("%d",&num);
                  key=create(num);
                  printf("\nEnter The Label:");
                  scanf("%s",l);
                  lprob(a,key,num);
                  printf("\nDo you want to Continue(y/n)?");
                  scanf(" %c",&ans);
                }
                display(a);
                break;
         case 2: search();
                  break;
      case 3: exit(0);    }
   }while(ch<=3);
}
```

Output :

```
Symbol Table Menu
1.Create a Symbol Table
2.Search in the Symbol Table
3.Exit
Enter your choice:1
Enter the Address:1024
Enter The Label:ABC
Do you want to Continue(y/n)?y
Enter the Address:2048
Enter The Label:EFG
Do you want to Continue(y/n)?y
Enter the Address:6144
Enter The Label:XYZ
Do you want to Continue(y/n)?n
The Symbol Table
*****************
Hash Values   Address        Label
0         0
1         1024   ABC
2         2048   EFG
3         0
4         0
5         0
6         6144   XYZ
7         0
8         0
9         0
10        0
Symbol Table Menu
1.Create a Symbol Table
2.Search in the Symbol Table
3.Exit
Enter your choice:2
Enter the Label:ABC
The Label --ABC-- is present in the Symbol Table at Address:1024
Symbol Table Menu
1.Create a Symbol Table
2.Search in the Symbol Table
3.Exit
Enter your choice:2
Enter the Label:PQR
The Label is Not Present in the Symbol Table!!
```

Symbol Table Menu
1.Create a Symbol Table
2.Search in the Symbol Table
3.Exit
Enter your choice:3