DINING PHILOSOPHER PROBLEM.

AIM:

TO WRITE A PROGRAM FOR DINNING PHILOSOPHER PROBLEM USING C LANGUAGE.

The dining philosophers problem is another classic synchronization problem which is used to evaluate situations where there is a need of allocating multiple resources to multiple processes.

the **dining philosophers problem** is an example problem often used in concurrent algorithm design to illustrate synchronization issues and techniques for resolving them.

ALGORITHM:

->we use the semaphore to solve the dining hilosopher problem which represents the fork.

->A fork can be picked by executing wait operation on semaphore and released by executing a single semaphore.

->Initially the element of fork are initialised to 1 as the forks are on the table and not picked by philisophers.

->the first wait operation is performed on fork[i] and fork[(i+1)%5].

->Which means that philosopher i has picked up the fork on his sides.Then eating action is performed.

->After that signal operation is performed on fork[i] and fork[(i+1)%5].That the philosopher i has ate and put down the fork on his sides.And he goes thinkging and this process will be continued.

Write a C program to implement dining philosopher's problem

For example:

Test	Input	Result
T1	5	the philosopher 1 falls hungry philosopher 1 can eat Eating in process.... philosopher 1 completed its works the philosopher 2 falls hungry philosopher 2 can eat Eating in process.... philosopher 2 completed its works the philosopher 3 falls hungry philosopher 3 can eat Eating in process.... philosopher 3 completed its works the philosopher 4 falls hungry philosopher 4 can eat Eating in process.... philosopher 4 completed its works the philosopher 5 falls hungry philosopher 5 can eat Eating in process.... philosopher 5 completed its works

PROGRAM:

```c
#include<stdio.h>
int takefork(int i,int left,int right,int s[])
{
if(s[i]==1&&s[left]!=2&&s[right]!=2)
{
s[i]=2;
printf(" philosopher %d can eat\n",i);
}
return 0;
}
int main()
{
int n,left,right;
scanf("%d",&n);
int s[6];
for(int i=1;i<=n;i++)
{
s[i]=0;
}
for(int i=1;i<=n;i++)
{
if(s[i]==0)
{
s[i]=1;
printf(" the philosopher %d falls hungry\n\n",i);
left=(i+4)%5;
right=(i+1)%5;
takefork(i,left,right,s);
if(s[i]==2)
{
printf(" Eating in process....\n");
s[i]=0;
printf(" philosopher %d completed its works\n\n",i);
takefork(left,(left+4)%5,(left+1)%5,s);
takefork(right,(right+4)%5,(right+1)%5,s);

}
}
}
}
```

OUTPUT:

	Test	Input	Expected	Got
✔	T1	5	the philosopher 1 falls hungry philosopher 1 can eat Eating in process.... philosopher 1 completed its works the philosopher 2 falls hungry philosopher 2 can eat Eating in process.... philosopher 2 completed its works the philosopher 3 falls hungry philosopher 3 can eat Eating in process.... philosopher 3 completed its works the philosopher 4 falls hungry philosopher 4 can eat Eating in process.... philosopher 4 completed its works the philosopher 5 falls hungry philosopher 5 can eat Eating in process.... philosopher 5 completed its works	the pl philo: Eatin philo: the pl philo: Eatin philo: the pl philo: Eatin philo: the pl philo: Eatin philo: the pl philo: Eatin philo:

Passed all tests! ✔

RESULT:

DINING PHILOSOPHER WAS EXCUTED SUCCESSFULLY USING C LANGAUGE.