AIM: To Write a C program to simulate the concept of Dining-Philosophers problem.

DESCRIPTION

The dining-philosophers problem is considered a classic synchronization problem because it is an example of a large class of concurrency-control problems. It is a simple representation of the need to allocate several resources among several processes in a deadlock-free and starvation-free manner. Consider five philosophers who spend their lives thinking and eating. The philosophers share a circular table surrounded by five chairs, each belonging to one philosopher. In the center of the table is a bowl of rice, and the table is laid with five single chopsticks. When a philosopher thinks, she does not interact with her colleagues. From time to time, a philosopher gets hungry and tries to pick up the two chopsticks that are closest to her (the chopsticks that are between her and her left and right neighbors). A philosopher may pick up only one chopstick at a time. Obviously, she cannot pick up a chopstick that is already in the hand of a neighbor. When a hungry philosopher has both her chopsticks at the same time, she eats without releasing her chopsticks. When she is finished eating, she puts down both of her chopsticks and starts thinking again. The dining-philosophers problem may lead to a deadlock situation and hence some rules have to be framed to avoid the occurrence of deadlock.

PROGRAM

```
int tph, philname[20], status[20], howhung, hu[20], cho;
main()
{
        int i; clrscr();
        printf("\n\nDINING PHILOSOPHER PROBLEM");
        printf("\nEnter the total no. of philosophers: ");
        scanf("%d",&tph);
        for(i=0;i<tph; i++)
        {
                philname[i]=(i+1);
                 status[i]=1;
        }
        printf("How many are hungry : ");
         scanf("%d", &howhung);
         if(howhung==tph)
        {
      printf("\n All are hungry..\nDead lock stage will occur");
      printf(\n"Exiting\n");
      else
      {
      for(i=0; i<howhung; i++)
      {
      printf("Enter philosopher %d position:",(i+1));
      scanf("%d",&hu[i]);
      status[hu[i]]=2;
      }
```

```
        do
                {
                        sprintf("1.One can eat at a time\t 2.Two can eat at a time
                        \t 3.Exit \n Enter yourchoice:");
                        scanf("%d", &cho);

                        switch(cho)
                        {
                        case 1:   one();
                                        break;
                      case 2: two();
                              break;
                    case 3: exit(0);

                                default: printf("\nInvalid option..");
                        }
                }while(1);
        }
}
one()
{

        int pos=0, x, i;
        printf("\nAllow one philosopher to eat at any time\n");
        for(i=0;i<howhung; i++, pos++)
        {
    printf("\n P %d is granted to eat", philname[hu[pos]]);
    for(x=pos; x<howhung; x++)
    printf("\nP %d is waiting", philname[hu[x]]);

        }
}

two()
{
        int i, j, s=0, t, r, x;
        printf("\n Allow two philosophers to eat at
        sametime\n");
        for(i=0; i<howhung; i++)
        {
```

```
                for(j=i+1;j<howhung;j++)
                {
                        if(abs(hu[i]-hu[j])>=1&&   abs(hu[
                        if(abs(hu[i]-hu[j])>=1 &&   abs(hu[i]-hu[j])!=4)

                        {
                              printf("\n\ncombination %d \n", (s+1));
                                t=hu[i];
                                r=hu[j];
                                s++;
                                printf("\nP %d and P %d are granted to eat", philname[hu[i]],
                                      philname[hu[j]]);



                                for(x=0;x<howhung;x++)
                                {
                                        if((hu[x]!=t) && (hu[x]!=r))
                                        printf("\nP %d is waiting", philname[hu[x]]);
                                }
                        }
                }
        }
}
```

INPUT

DINING PHILOSOPHER PROBLEM
Enter the total no. of philosophers: 5 How many
are hungry : 3
Enter philosopher 1 position: 2
Enter philosopher 2 position: 4
Enter philosopher 3 position: 5

OUTPUT

1. One can eat at a time
2. Two can eat at a time
3. Exit Enter your choice: 1

Allow one philosopher to eat at any time
P 3 is granted to eat
P 3 is waiting
P 5 is waiting
P 0 is waiting
P 5 is granted to eat
P 5 is waiting
P 0 is waiting
P 0 is granted to eat
P 0 is waiting

1.One can eat at a time

2.Two can eat at a time

3.Exit

Enter your choice: 2

Allow two philosophers to eat at same time
combination 1
P 3 and P 5 are granted to eat
P 0 is waiting

combination 2
P 3 and P 0 are granted to eat
P 5 is waiting

combination 3
P 5 and P 0 are granted to eat
P 3 is waiting

1.One can eat at a time
2.Two can eat at a time
3.Exit
Enter your choice: 3