~~Q Aim: To write a C program to simulate producer- consumer problem using semaphore~~

~~Discription~~

~~producer consumer problem is a ~~synchronizes~~ synchronization problem there is fixed size buffer where~~

Q Aim! To write a c program to simulate the concept of Dining -philosophers problem.

~~Ds~~ Description

The dining -philosophers problem is

program

```
int tph, philname[20], status[20], how
hung, hu[20], cho;
int main()
{
int i;
printf("\n\n dining philosopher problem");
printf("\n enter 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 \n Deadlock
    Stage will occur");
  printf (\n" Exiting \n");
   else
  {
 for (i = 0; i < howhung; i++)
  {
  printf (~~\n" Exi~~ " Enter philosopher %
         d position:", (i+1));
      Scanf ("%d", &hu[i]);
        Status[hu[i]] = 2;
      }
    do
   {
  printf ("1. one can eat at a time \t2.
         Two can eat a time
         \t3. Exit \n Enter your
         choice : ");
  Scanf ("%d", &ch);
```

```
Switch(cho)
{
case 1: one();
case 2: two();
~~break~~ case 3: exit(0);
          break;
        }
       }
       one()
      {
      }
    while (1);
default: printf("\n invalid option...");

int pos = 0, x, i;
printf("\n Allow one Philosopher
        to eat at any time \n");
for (i=0; i<howhung; i++, pos++)
{
printf("\n Allow 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 ("\n P%d is waiting", philname[hu
                                   [x]]);
 }
 two()
{
}
int i, j s=0 , t, r, x;
printf(" \n allow two philosophers to eat at
        same time \n");
  for (i=0 ; i< how hung ; i++)
   {
    for (j = i+1; j< how hung ; i++)
    {
    if (abs (hu[i] - hu [j]) >=1 && abs
                                (hu[i] -
                                 hu[j]!
                                  =4)
    {
     printf("\n\n combination%d \n",
              (s+1));
              t = hu[i];
            r = hu [j];
              s ++;
Printf (" \n P%d and P%d are granted
         to eat", philname [hu[i]],
                    philname[hu[j]]);
for (x=0;x< how hung; x++)
  {
   if((hu [x]! =t) && (hu [x]! =r))
   printf("\n P %d is waiting", Philname[hu
   ]);
   }
  }
 }
}
}
```

Out Put

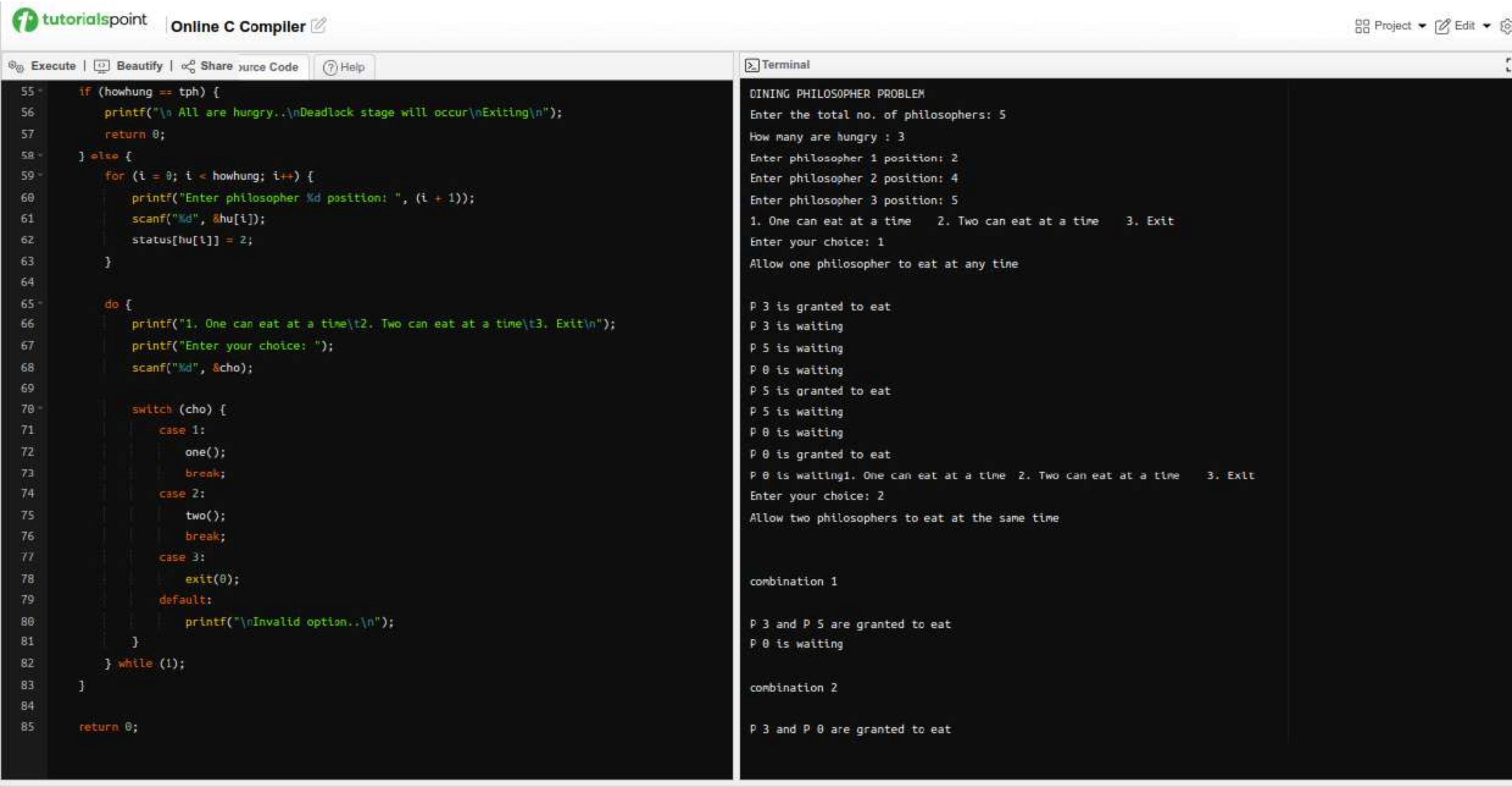
```
~~Dinner~~ Dining philosopher problem
  Enter the total no of philosophers : 5
How many are  hungry : 3
  Enter philosopher 1 position: 2
  Enter philosopher 2  position: 4
   Enter philosopher 3  position: 5

1) one can eat a time  2.) Two can eat a time 3.

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
```



```c
    if (howhung == tph) {
        printf("\n All are hungry..\nDeadlock stage will occur\nExiting\n");
        return 0;
    } else {
        for (i = 0; i < howhung; i++) {
            printf("Enter philosopher %d position: ", (i + 1));
            scanf("%d", &hu[i]);
            status[hu[i]] = 2;
        }

        do {
            printf("1. One can eat at a time\t2. Two can eat at a time\t3. Exit\n");
            printf("Enter your choice: ");
            scanf("%d", &cho);

            switch (cho) {
                case 1:
                    one();
                    break;
                case 2:
                    two();
                    break;
                case 3:
                    exit(0);
                default:
                    printf("\nInvalid option..\n");
            }
        } while (1);
    }

    return 0;
```

```
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
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 waiting1. 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 the 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
```