**Practical – 11: To implement Dinning Philosophers Problem.**

#include <stdio.h>

#define n 4

// BE20F05F062 Akash Shridharan

int compltedPhilo = 0, i;

struct fork

{

int taken;

} ForkAvil[n];

struct philosp

{

int left;

int right;

} Philostatus[n];

void goForDinner(int philID)

{

if (Philostatus[philID].left == 10 && Philostatus[philID].right == 10)

printf("Philosopher %d completed his dinner\n", philID + 1);

else if (Philostatus[philID].left == 1 && Philostatus[philID].right == 1)

{

printf("Philosopher %d completed his dinner\n", philID + 1);

Philostatus[philID].left = Philostatus[philID].right = 10;

int otherFork = philID - 1;

if (otherFork == -1)

otherFork = (n - 1);

ForkAvil[philID].taken = ForkAvil[otherFork].taken = 0;

printf("Philosopher %d released fork %d and fork %d\n", philID + 1, philID + 1, otherFork + 1);

compltedPhilo++;

}

else if (Philostatus[philID].left == 1 && Philostatus[philID].right == 0)

{

if (philID == (n - 1))

{

if (ForkAvil[philID].taken == 0)

{

ForkAvil[philID].taken = Philostatus[philID].right = 1;

printf("Fork %d taken by philosopher %d\n", philID + 1, philID + 1);

}

else

{

printf("Philosopher %d is waiting for fork %d\n", philID + 1, philID + 1);

}

}

else

{

int dupphilID = philID;

philID -= 1;

if (philID == -1)

philID = (n - 1);

if (ForkAvil[philID].taken == 0)

{

ForkAvil[philID].taken = Philostatus[dupphilID].right = 1;

printf("Fork %d taken by Philosopher %d\n", philID + 1, dupphilID + 1);

}

else

{

printf("Philosopher %d is waiting for Fork %d\n", dupphilID + 1, philID + 1);

}

}

}

else if (Philostatus[philID].left == 0)

{

if (philID == (n - 1))

{

if (ForkAvil[philID - 1].taken == 0)

{

ForkAvil[philID - 1].taken = Philostatus[philID].left = 1;

printf("Fork %d taken by philosopher %d\n", philID, philID + 1);

}

else

{

printf("Philosopher %d is waiting for fork %d\n", philID + 1, philID);

}

}

else

{

if (ForkAvil[philID].taken == 0)

{

ForkAvil[philID].taken = Philostatus[philID].left = 1;

printf("Fork %d taken by Philosopher %d\n", philID + 1, philID + 1);

}

else

{

printf("Philosopher %d is waiting for Fork %d\n", philID + 1, philID + 1);

}

}

}

else{}

}

int main()

{

for (i = 0; i < n; i++)

ForkAvil[i].taken = Philostatus[i].left = Philostatus[i].right = 0;

while (compltedPhilo < n)

{

for (i = 0; i < n; i++)

goForDinner(i);

printf("\nTill now num of philosophers completed dinner are %d\n\n", compltedPhilo);

}

return 0;

}

**Output :-**

Fork 1 taken by Philosopher 1

Fork 2 taken by Philosopher 2

Fork 3 taken by Philosopher 3

Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 0

Fork 4 taken by Philosopher 1

Philosopher 2 is waiting for Fork 1

Philosopher 3 is waiting for Fork 2

Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 0

Philosopher 1 completed his dinner

Philosopher 1 released fork 1 and fork 4

Fork 1 taken by Philosopher 2

Philosopher 3 is waiting for Fork 2

Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 1

Philosopher 1 completed his dinner

Philosopher 2 completed his dinner

Philosopher 2 released fork 2 and fork 1

Fork 2 taken by Philosopher 3

Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 2

Philosopher 1 completed his dinner

Philosopher 2 completed his dinner

Philosopher 3 completed his dinner

Philosopher 3 released fork 3 and fork 2

Fork 3 taken by philosopher 4

Till now num of philosophers completed dinner are 3

Philosopher 1 completed his dinner

Philosopher 2 completed his dinner

Philosopher 3 completed his dinner

Fork 4 taken by philosopher 4

Till now num of philosophers completed dinner are 3

Philosopher 1 completed his dinner

Philosopher 2 completed his dinner

Philosopher 3 completed his dinner

Philosopher 4 completed his dinner

Philosopher 4 released fork 4 and fork 3

Till now num of philosophers completed dinner are 4