Lab 4:

1. Write a C program to simulate producer-consumer problem using semaphores.

#include <stdio.h>

#include <stdlib.h>

int mutex = 1;

int full = 0;

int empty = 2;

int x = 0;

int wait(int s) {

return (--s);

}

int signal(int s) {

return (++s);

}

void producer() {

mutex = wait(mutex);

full = signal(full);

empty = wait(empty);

x++;

printf("Producer produces item %d\n", x);

mutex = signal(mutex);

}

void consumer() {

mutex = wait(mutex);

full = wait(full);

empty = signal(empty);

printf("Consumer consumes item %d\n", x);

x--;

mutex = signal(mutex);

}

int main() {

int choice;

while (1) {

printf("\n1. Producer\n2. Consumer\n3. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

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;

case 3:

exit(0);

default:

printf("Invalid choice!\n");

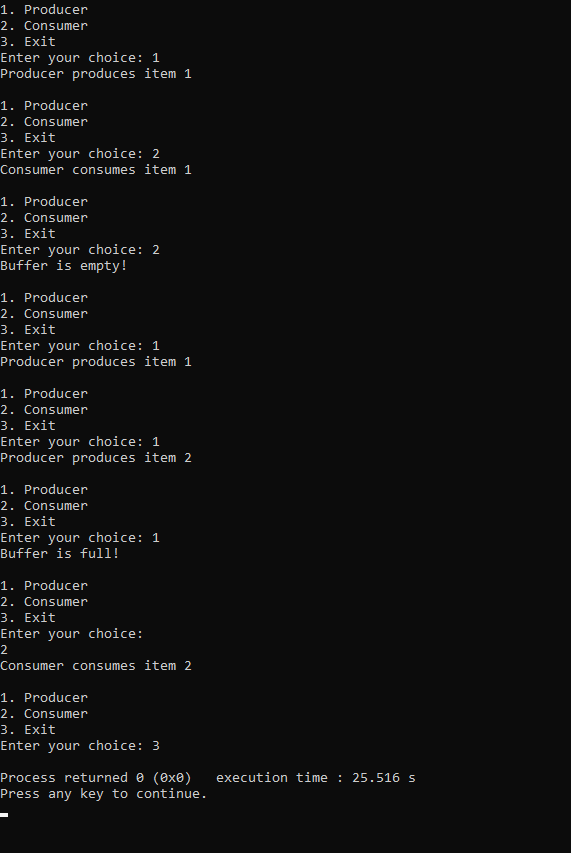
}

}

return 0

}

Output:



1. Write a C program to simulate the concept of Dining-Philosopher’s problem.

#include <stdio.h>

#include <stdlib.h>

#define MAX 10

int totalPhilosophers;

int hungry[MAX];

int areNeighbors(int a, int b) {

    return (abs(a - b) == 1 || abs(a - b) == totalPhilosophers - 1);

}

void option1(int count) {

    printf("\nAllow one philosopher to eat at any time\n");

    for (int i = 0; i < count; i++) {

        printf("P %d is granted to eat\n", hungry[i]);

        for (int j = 0; j < count; j++) {

            if (j != i) {

                printf("P %d is waiting\n", hungry[j]);

            }

        }

    }

}

void option2(int count) {

    printf("\nAllow two philosophers to eat at same time\n");

    int combination = 1;

    for (int i = 0; i < count; i++) {

        for (int j = i + 1; j < count; j++) {

            if (!areNeighbors(hungry[i], hungry[j])) {

                printf("combination %d\n", combination++);

                printf("P %d and P %d are granted to eat\n", hungry[i], hungry[j]);

                for (int k = 0; k < count; k++) {

                    if (k != i && k != j) {

                        printf("P %d is waiting\n", hungry[k]);

                    }

                }

                printf("\n");

            }

        }

    }

    if (combination == 1) {

        printf("No combinations found where two non-neighbor philosophers can eat.\n");

    }

}

int main() {

    int hungryCount;

    printf("DINING PHILOSOPHER PROBLEM\n");

    printf("Enter the total no. of philosophers: ");

    scanf("%d", &totalPhilosophers);

    printf("How many are hungry: ");

    scanf("%d", &hungryCount);

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

        printf("Enter philosopher %d position: ", i + 1);

        scanf("%d", &hungry[i]);

    }

    int choice;

    do {

        printf("\n1. One can eat at a time   2. Two can eat at a time   3. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

        case 1:

            option1(hungryCount);

            break;

        case 2:

            option2(hungryCount);

            break;

        case 3:

            printf("Exiting...\n");

            break;

        default:

            printf("Invalid choice!\n");

        }

    } while (choice != 3);

    return 0;

Output:

