MARRI LAXMAN REDDY INSTITUTE OF TECHNOLOGY AND MANAGEMENT

INDEX

SI. No.	Date	Name of the Experiment	Remarks
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			

WEEK - 6	**Exp. No.:** 1 **Date:** - -

AIM: Write a C program to implement the Producer – Consumer problem using semaphores using UNIX/LINUX system calls.

Program:

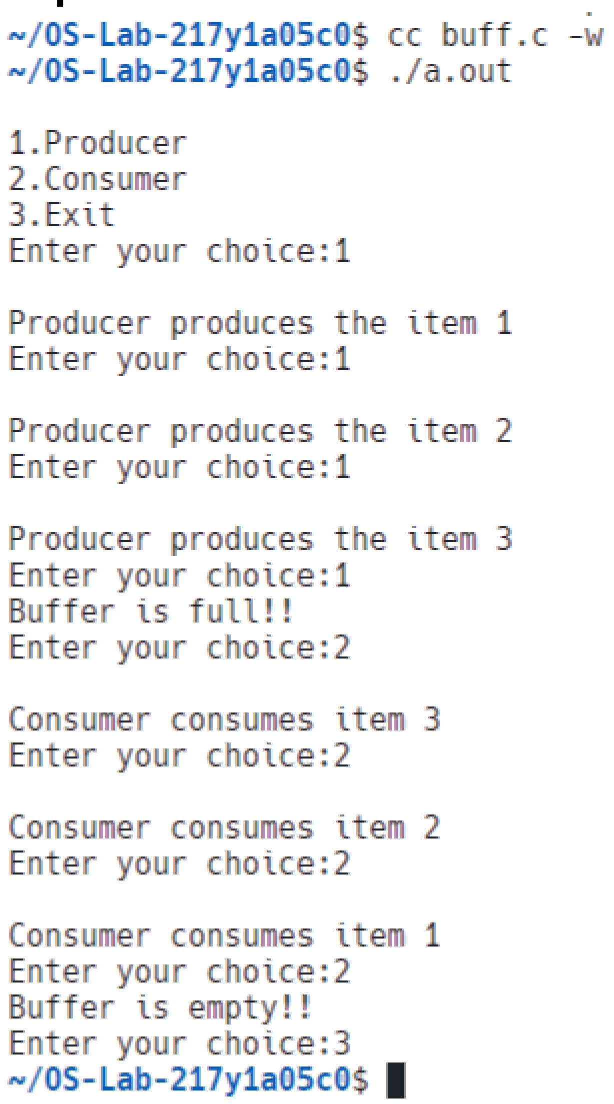
```c
#include <stdio.h>
#include <stdlib.h>
int mutex = 1, full = 0, empty = 3, x = 0;
int main() {
  int n;
  void producer();
  void consumer();
  int wait(int);
  int signal(int);
  printf("\n1.Producer\n2.Consumer\n3.Exit");
  while (1) {
    printf("\nEnter your choice:");
    scanf("%d", &n);

    switch (n) {
    case 1:
      if ((mutex == 1) && (empty != 0))
        producer();
      else
        printf("Buffer is full!!");
      break;
    case 2:
      if ((mutex == 1) && (full != 0))
        consumer();
      else
        printf("Buffer is empty!!");
      break;
    case 3: exit(0);
    }
  }
  return 0;
}
int wait(int s) { return (--s); }
int signal(int s) { return (++s); }
void producer() {
  mutex = wait(mutex);
  full = signal(full);
  empty = wait(empty);
  printf("\nProducer produces the item %d", ++x);

  mutex = signal(mutex);
}
void consumer() {
  mutex = wait(mutex);
  full = wait(full);
  empty = signal(empty);
  printf("\nConsumer consumes item %d", x--);
  mutex = signal(mutex);
}
```

WEEK - 6	Exp. No.: 2 Date: -09-2-23

Output:



```
~/OS-Lab-217y1a05c0$ cc buff.c -w
~/OS-Lab-217y1a05c0$ ./a.out

1.Producer
2.Consumer
3.Exit
Enter your choice:1

Producer produces the item 1
Enter your choice:1

Producer produces the item 2
Enter your choice:1

Producer produces the item 3
Enter your choice:1
Buffer is full!!
Enter your choice:2

Consumer consumes item 3
Enter your choice:2

Consumer consumes item 2
Enter your choice:2

Consumer consumes item 1
Enter your choice:2
Buffer is empty!!
Enter your choice:3
~/OS-Lab-217y1a05c0$ █
```

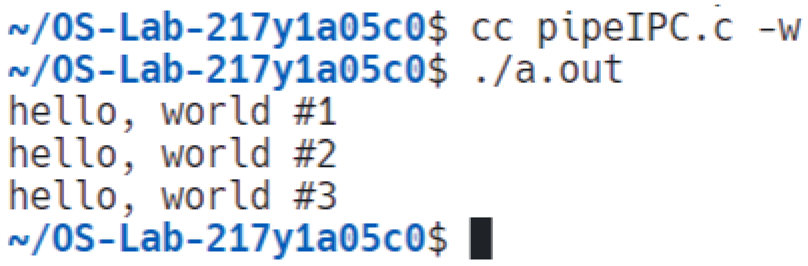
WEEK - 7	**Exp. No.:** 1 **Date:** - -

AIM: Write a C program to illustrate the Pipes IPC mechanism.

Program:

```
#include <stdio.h>
#include <unistd.h>
#define MSGSIZE 16
char *msg1 = "hello, world #1";
char *msg2 = "hello, world #2";
char *msg3 = "hello, world #3";
int main() {
  char inbuf[MSGSIZE];
  int p[2], i;
  if (pipe(p) < 0)
    exit(1);
  /* continued */
  /* write pipe */
  write(p[1], msg1, MSGSIZE);
  write(p[1], msg2, MSGSIZE);
  write(p[1], msg3, MSGSIZE);
  for (i = 0; i < 3; i++) {
    /* read pipe */
    read(p[0], inbuf, MSGSIZE);
    printf("% s\n", inbuf);
  }
  return 0;
}
```

Output:



```
~/OS-Lab-217y1a05c0$ cc pipeIPC.c -w
~/OS-Lab-217y1a05c0$ ./a.out
hello, world #1
hello, world #2
hello, world #3
~/OS-Lab-217y1a05c0$ █
```

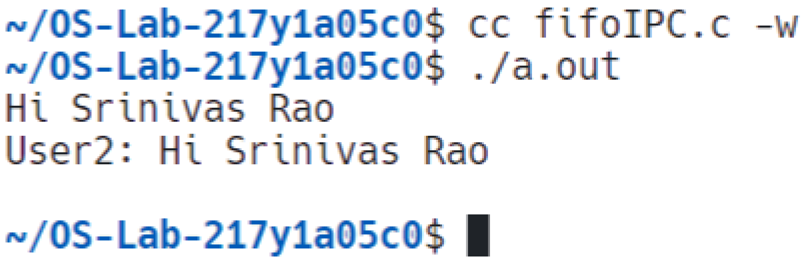
WEEK - 8	**Exp. No.:** 1 **Date:** - -

AIM: Write a C program to illustrate the FIFO IPC mechanism.

Program:

```
#include <fcntl.h>
#include <stdio.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
int main() {
  int fd;
  char *myfifo = "/tmp/myfifo";
  mkfifo(myfifo, 0666);
  char arr1[80], arr2[80];
  while (1) {
    fd = open(myfifo, O_WRONLY);
    fgets(arr2, 80, stdin);
    write(fd, arr2, strlen(arr2) + 1);
    close(fd);
    fd = open(myfifo, O_RDONLY);
    read(fd, arr1, sizeof(arr1));
    printf("User2: %s\n", arr1);
    close(fd);
  }
  return 0;
}
```

Output:



```
~/OS-Lab-217y1a05c0$ cc fifoIPC.c -w
~/OS-Lab-217y1a05c0$ ./a.out
Hi Srinivas Rao
User2: Hi Srinivas Rao

~/OS-Lab-217y1a05c0$ 
```

WEEK - 8	Exp. No.: 2 Date: - -

AIM: Write a C program to illustrate the Message Queue IPC mechanism..

Program 1:

```c
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
struct mesg_buffer {
  long mesg_type;
  char mesg_text[100];
} message;
int main() {
  key_t key;
  int msgid;
  key = ftok("progfile", 65);
  msgid = msgget(key, 0666 | IPC_CREAT);
  message.mesg_type = 1;
  printf("Write Data : ");
  fgets(message.mesg_text, sizeof(message.mesg_text), stdin);
  msgsnd(msgid, &message, sizeof(message), 0);
  printf("Data send is : %s \n", message.mesg_text);
  return 0;
}
```

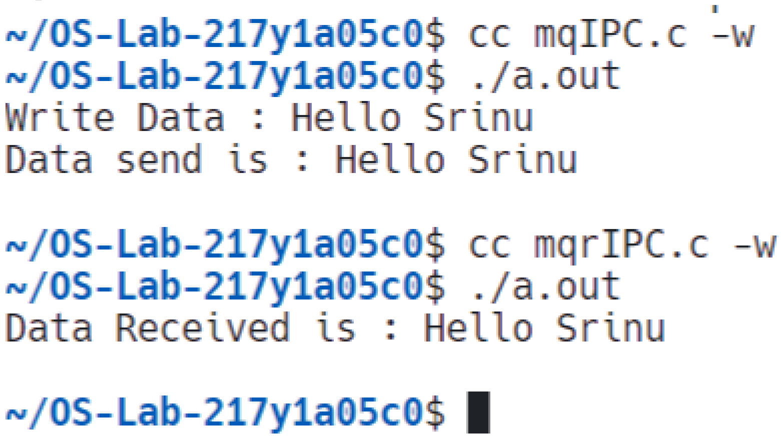
Program 2:

```c
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
struct mesg_buffer {
  long mesg_type;
  char mesg_text[100];
} message;
int main() {
  key_t key;

  int msgid;
  key = ftok("progfile", 65);
  msgid = msgget(key, 0666 | IPC_CREAT);
  msgrcv(msgid, &message, sizeof(message), 1, 0);
  printf("Data Received is : %s \n", message.mesg_text);
  msgctl(msgid, IPC_RMID, NULL);
  return 0;
}
```

WEEK - 8	Exp. No.: 3 Date: - -

Output:



```
~/OS-Lab-217y1a05c0$ cc mqIPC.c -w
~/OS-Lab-217y1a05c0$ ./a.out
Write Data : Hello Srinu
Data send is : Hello Srinu

~/OS-Lab-217y1a05c0$ cc mqrIPC.c -w
~/OS-Lab-217y1a05c0$ ./a.out
Data Received is : Hello Srinu

~/OS-Lab-217y1a05c0$ █
```

WEEK - 8	**Exp. No.:** 4 **Date:** - -

AIM: Write a C program to illustrate the Shared Memory IPC mechanism

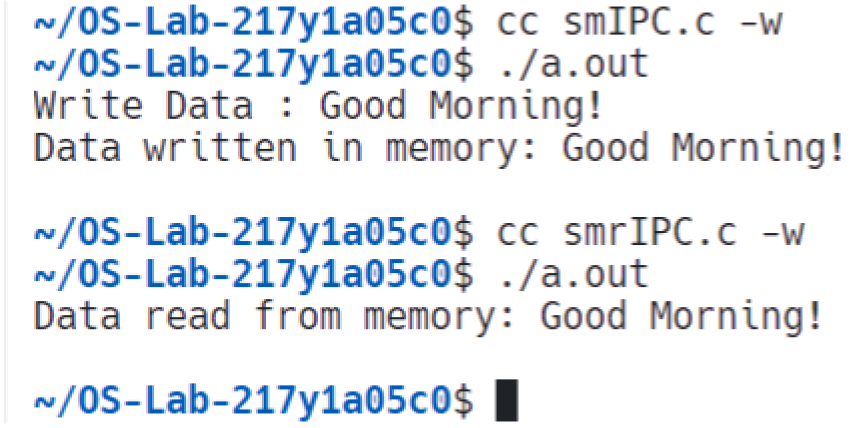
Program 1:

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/shm.h>
int main() {
  key_t key = ftok("shmfile", 65);
  int shmid = shmget(key, 1024, 0666 | IPC_CREAT);
  char *str = (char *)shmat(shmid, (void *)0, 0);
  printf("Write Data : ");
  fgets(str, 1024, stdin);
  printf("Data written in memory: %s\n", str);
  shmdt(str);
  return 0;
}
```

Program 2:

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/shm.h>
int main() {
  key_t key = ftok("shmfile", 65);
  int shmid = shmget(key, 1024, 0666 | IPC_CREAT);
  char *str = (char *)shmat(shmid, (void *)0, 0);
  printf("Data read from memory: %s\n", str);
  shmdt(str);
  shmctl(shmid, IPC_RMID, NULL);
  return 0;
}
```

Output:



```
~/OS-Lab-217y1a05c0$ cc smIPC.c -w
~/OS-Lab-217y1a05c0$ ./a.out
Write Data : Good Morning!
Data written in memory: Good Morning!

~/OS-Lab-217y1a05c0$ cc smrIPC.c -w
~/OS-Lab-217y1a05c0$ ./a.out
Data read from memory: Good Morning!

~/OS-Lab-217y1a05c0$ █
```

WEEK - 9	**Exp. No.:** 1 **Date:** - -

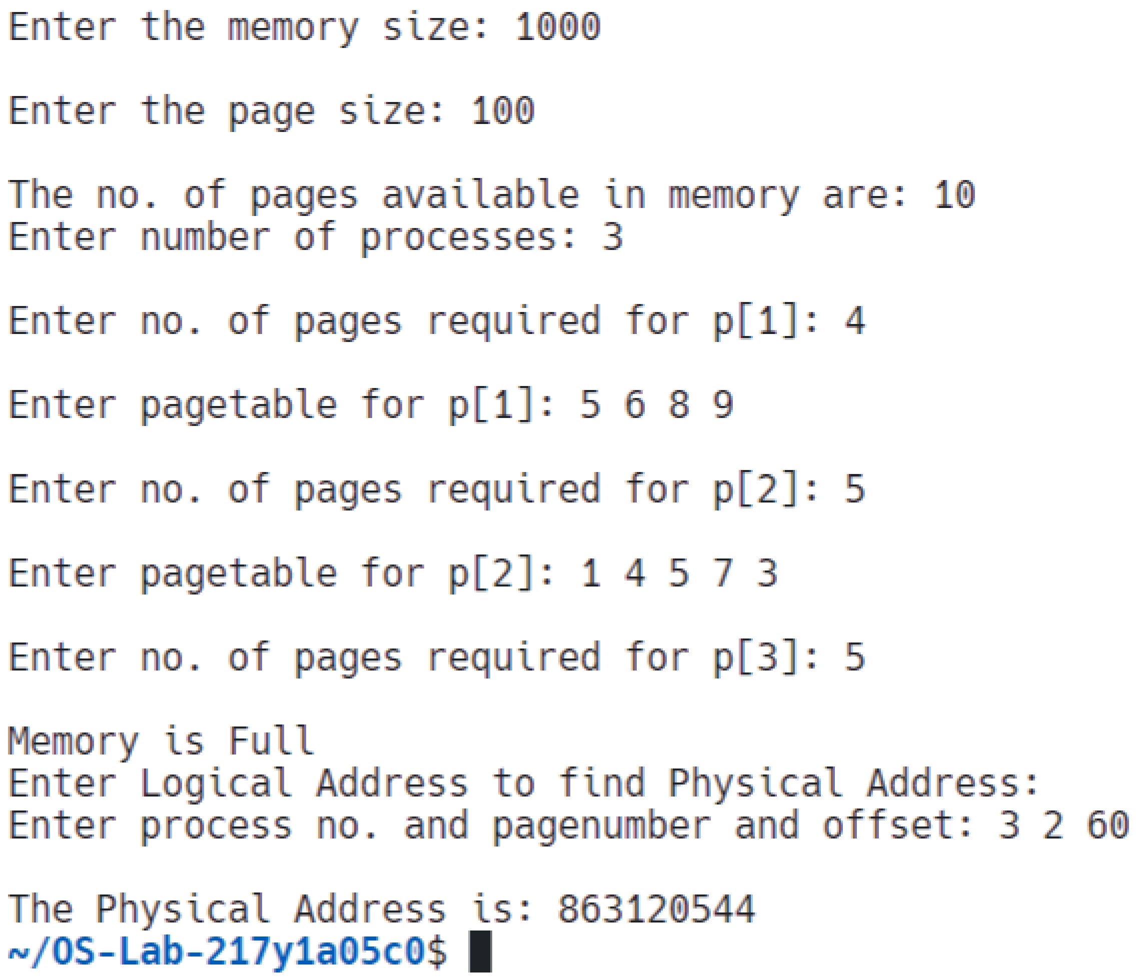
AIM: Write a C program to simulate paging technique of memory management.

Program:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int ms, ps, nop, np, rempages, i, j, x, y, pa, offset;
  int s[10], fno[10][20];
  system("clear");
  printf("\nEnter the memory size: ");
  scanf("%d", &ms);
  printf("\nEnter the page size: ");
  scanf("%d", &ps);
  nop = ms / ps;
  printf("\nThe no. of pages available in memory are: %d ", nop);
  printf("\nEnter number of processes: ");
  scanf("%d", &np);
  rempages = nop;
  for (i = 1; i <= np; i++) {
    printf("\nEnter no. of pages required for p[%d]: ", i);
    scanf("%d", &s[i]);
    if (s[i] > rempages) {
      printf("\nMemory is Full");
      break;
    }
    rempages = rempages - s[i];
    printf("\nEnter pagetable for p[%d]: ", i);
    for (j = 0; j < s[i]; j++)
      scanf("%d", &fno[i][j]);
  }
  printf("\nEnter Logical Address to find Physical Address: ");
  printf("\nEnter process no. and pagenumber and offset: ");
  scanf("%d %d %d", &x, &y, &offset);
  if (x > np || y >= s[i] || offset >= ps)
    printf("\nInvalid Process or Page Number or offset:");
  else {
    pa = fno[x][y] * ps + offset;
    printf("\nThe Physical Address is: %d", pa);
  }printf("\n");
  getchar();
}
```

WEEK - 9	**Exp. No.:** 2 **Date:** - -

Output:



```
Enter the memory size: 1000

Enter the page size: 100

The no. of pages available in memory are: 10
Enter number of processes: 3

Enter no. of pages required for p[1]: 4

Enter pagetable for p[1]: 5 6 8 9

Enter no. of pages required for p[2]: 5

Enter pagetable for p[2]: 1 4 5 7 3

Enter no. of pages required for p[3]: 5

Memory is Full
Enter Logical Address to find Physical Address:
Enter process no. and pagenumber and offset: 3 2 60

The Physical Address is: 863120544
~/OS-Lab-217y1a05c0$ █
```

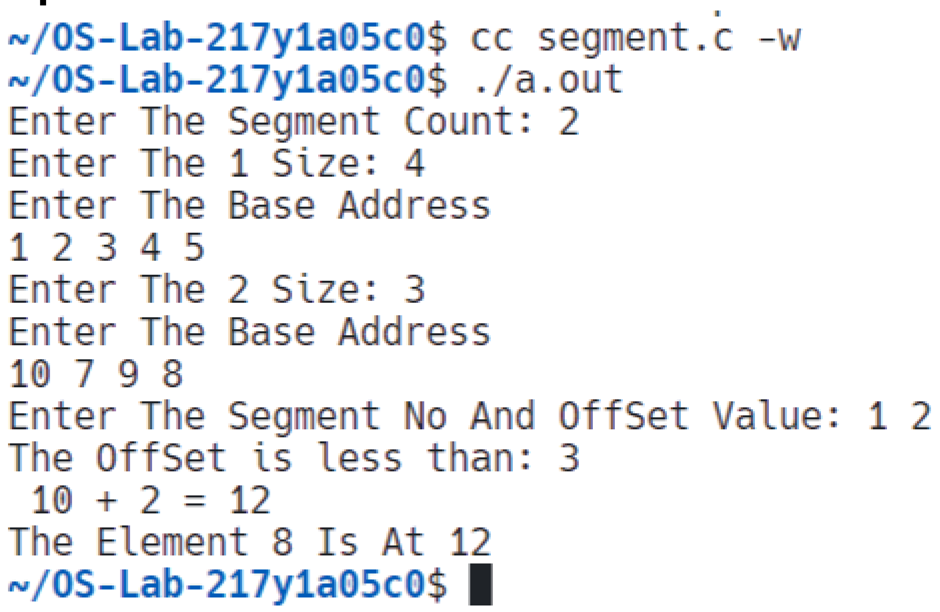
WEEK - 10	Exp. No.: 1 Date: - -

AIM: AIM:Write a c program to simulate segmentation technique of memory management.

Program:

```
#include <stdio.h>
int main() {
  int a[100][100], b[1000], i, j, n, x, base, size, seg, off;
  printf("Enter The Segment Count: ");
  scanf("%d", &n);
  for (i = 0; i < n; i++) {
    printf("Enter The %d Size: ", i + 1);
    scanf("%d", &size);
    a[i][0] = size;
    printf("Enter The Base Address\n");
    scanf("%d", &base);
    a[i][i] = base;
    for (j = 0; j < size; j++) {
      x = 0;
      scanf("%d", &x);
      b[base] = x;
      base++;
      b[base] = x;
    }
  }
  printf("Enter The Segment No And OffSet Value: ");
  scanf("%d %d", &seg, &off);
  if (off < a[seg][0]) {
    int abs = a[seg][1] + off;
    printf("The OffSet is less than: %d", a[seg][0]);
    printf("\n %d + %d = %d\n", a[seg][1], off, abs);
    printf("The Element %d Is At %d\n", b[abs + 1], abs);
  } else {
    printf("Error in Locating");
  }
}
```

Output:



```
~/OS-Lab-217y1a05c0$ cc segment.c -w
~/OS-Lab-217y1a05c0$ ./a.out
Enter The Segment Count: 2
Enter The 1 Size: 4
Enter The Base Address
1 2 3 4 5
Enter The 2 Size: 3
Enter The Base Address
10 7 9 8
Enter The Segment No And OffSet Value: 1 2
The OffSet is less than: 3
 10 + 2 = 12
The Element 8 Is At 12
~/OS-Lab-217y1a05c0$ 
```

WEEK - 11	**Exp. No.:** **Date:** - -

AIM: Write a c program to simulate the concept of Dining-philosophers problem.

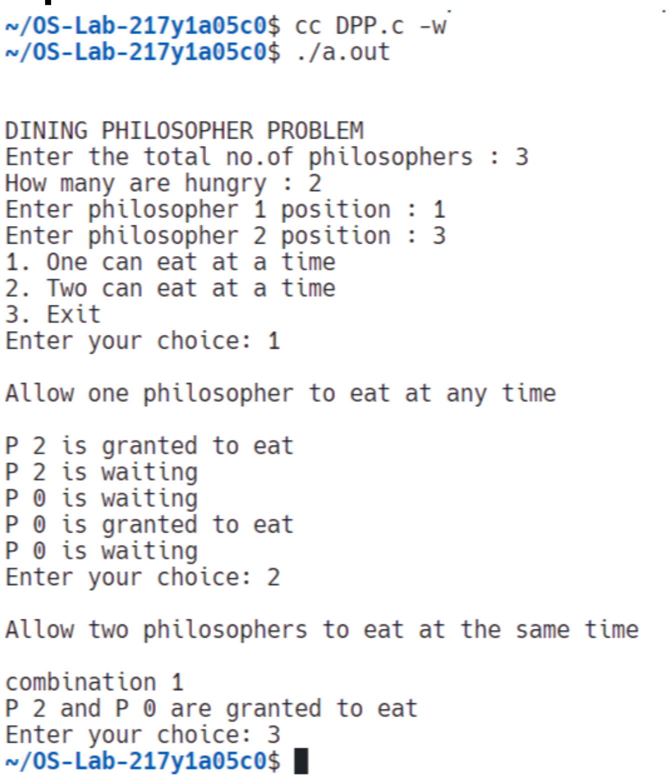
Program:

```
#include <stdio.h>
#include <stdlib.h>
int tph, philname[20], status[20], howhung, hu[20], cho;
void one();
void two();
int main() {
  int i;
  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("\nAll are hungry..\nDeadlock stage will occur");
    printf("\nExiting..");
  } else {
    for (i = 0; i < howhung; i++) {
      printf("Enter philosopher %d position : ", (i + 1));
      scanf("%d", &hu[i]);
      status[hu[i]] = 2;
    }
    printf("1. One can eat at a time\n2. Two can eat at a time\n3. Exit");
    do {
      printf("\nEnter your choice: ");
      scanf("%d", &cho);
      switch (cho) {
      case 1: one();
        break;
      case 2: two();
        break;
      case 3: exit(0);
      default: printf("\nInvalid option..");
      }
    } while (1);
  }
  return 0;
}
```

WEEK - 11	Exp. No.: Date: - -

```c
void one() {
  int pos = 0, x, i;
  printf("\nAllow one philosopher to eat at any time\n");
  for (i = 0; i < howhung; i++, pos++) {
    printf("\nP %d is granted to eat", philname[hu[pos]]);
    for (x = pos; x < howhung; x++)
      printf("\nP %d is waiting", philname[hu[x]]);
  }
}
void two() {
  int i, j, s = 0, t, r, x;
  printf("\nAllow two philosophers to eat at the same time\n");
  for (i = 0; i < howhung; i++) {
    for (j = i + 1; j < howhung; j++) {
      if (abs(hu[i] - hu[j]) >= 1 && abs(hu[i] - hu[j]) != 4) {
        printf("\ncombination %d\n", (s + 1));
        t = hu[i];
        r = hu[j];
        s++;
        printf("P %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]]);
        }
      }
    }
  }
}
```

Output:



```
~/OS-Lab-217y1a05c0$ cc DPP.c -w
~/OS-Lab-217y1a05c0$ ./a.out


DINING PHILOSOPHER PROBLEM
Enter the total no.of philosophers : 3
How many are hungry : 2
Enter philosopher 1 position : 1
Enter philosopher 2 position : 3
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 2 is granted to eat
P 2 is waiting
P 0 is waiting
P 0 is granted to eat
P 0 is waiting
Enter your choice: 2

Allow two philosophers to eat at the same time

combination 1
P 2 and P 0 are granted to eat
Enter your choice: 3
~/OS-Lab-217y1a05c0$ █
```