```
// Online C compiler to run C program online
#include <stdio.h>
#include <malloc.h>
#include<string.h>

/* * Node Declaration */
struct person
{
   char name[50];
   struct person *next;
};

struct ticket
{
char number[50];
struct ticket *next;
};

struct person* create_person(char *);
struct ticket* create_ticket(char *);

void insert_person();
void insert_ticket();
void display_persons();
void display_tickets();

struct person *newperson;
struct person *ptrPerson;
struct ticket *newticket;
struct ticket *ptrTicket;

struct person * firstPerson = NULL;
struct person * lastPerson = NULL;
struct ticket * firstTicket = NULL;
struct ticket * lastTicket = NULL;

/* * Main :contains menu */
int main()
{
int ch;
char ans = 'Y';

while (ans == 'Y'||ans == 'y')
{
```

```
printf("\n--------------------------------\n");
printf("\nOperations on Ticket & Queue\n");
printf("\n--------------------------------\n");
printf("\n1. Add Person in Q");
        printf("\n2. Add ticket");
printf("\n3.Display All Persons in Q:");
        printf("\n4.Display All Tickets");
printf("\n10.Exit\n");
printf("\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
printf("\nEnter your choice : ");
scanf("%d", &ch);

switch (ch)
{
case 1:
   printf("\n...Adding Person in Q...\n");
   insert_person();
   break;

  case 2:
   printf("\n...Adding Ticket...\n");
   insert_ticket();
   break;
case 3:
   printf("\n...Displaying Persons in Q From Beginning to End : \n");
   display_persons();
   break;
case 4:
   printf("\n...Displaying Tickets From Beginning to End :\n");
   display_tickets();
   break;

case 10:
   printf("\n...Exiting...\n");
   return 0;
   break;
default:
   printf("\n...Invalid Choice...\n");
   break;
}
printf("\n\n You want to continue (Y/N)");
scanf(" %c", &ans);
}
return 0;
```

```
 }

/* * Creating Node */
struct person* create_person(char *ptrName)
{
newperson = (struct person *)malloc(sizeof(struct person));
if (newperson == NULL)
{
printf("\nMemory was not allocated");
return 0;
}
else
{
strcpy(newperson->name, ptrName);
newperson->next = NULL;
return newperson;
}
}

struct ticket* create_ticket(char *tktNumber)
{
newticket = (struct ticket *)malloc(sizeof(struct ticket));
if (newticket == NULL)
{
printf("\nMemory was not allocated");
return 0;
}
else
{
strcpy(newticket->number, tktNumber);
newticket->next = NULL;
return newticket;
}
}

/* * Inserting Node at Last */
void insert_person()
{
char val[50];

printf("\nEnter the name of person : ");
scanf("%s", val);
newperson = create_person(val);
if (firstPerson == lastPerson && lastPerson == NULL)
```

```
{
firstPerson = lastPerson = newperson;
firstPerson->next = NULL;
lastPerson->next = NULL;
}
else
{
lastPerson->next = newperson;
lastPerson = newperson;
lastPerson->next = NULL;
}
printf("\n----INSERTED----");
}

void insert_ticket()
{
char val[50];

printf("\nEnter the number of ticket : ");
scanf("%s", val);
newticket = create_ticket(val);
if (firstTicket == lastTicket && lastTicket == NULL)
{
firstTicket = lastTicket = newticket;
firstTicket->next = NULL;
lastTicket->next = NULL;
}
else
{
lastTicket->next = newticket;
lastTicket = newticket;
lastTicket->next = NULL;
}
 printf("\n----INSERTED----");
}


/* * Displays non-empty List from Beginning to End */
void display_persons()
{
if (firstPerson == NULL)
{

printf(":No Person in the list to display\n");
```

```
}
else
{
for (ptrPerson = firstPerson; ptrPerson != NULL;ptrPerson = ptrPerson->next)
{
  printf("%s\t", ptrPerson->name);
}
}
}

void display_tickets()
{
if (firstTicket == NULL)
{
printf(":No Ticket in the list to display\n");
}
else
{
for (ptrTicket = firstTicket; ptrTicket != NULL;ptrTicket = ptrTicket->next)
{
    printf("%s\t", ptrTicket->number);
}
}
}
```