Structures – Q1 (computeCircle)

A structure called circle is defined below. The structure consists of the radius of the circle and the (x,y) coordinates of its centre. A structure called circle is defined below. The structure consists of the radius of the circle and the (x,y) coordinates of its centre.

```
struct circle {
    double radius;
    double x;
    double y;
};
```

(a) Implement the function **intersect()** that returns 1 if two circles intersect, and 0 otherwise. Two circles intersect when the distance between their centres is less than or equal to the sum of their radii. The function prototype is given below:

```
int intersect(struct circle c1, struct circle c2);
```

(b) Implement the function **contain()** that returns 1 if *c1* contains *c2*, i.e. circle *c2* is found inside circle *c1*. Otherwise, the function returns 0. Circle *c1* contains circle *c2* when the radius of *c1* is larger than or equal to the sum of the radius of *c2* and the distance between the centres of *c1* and *c2*. The function prototype of contain() is given below:

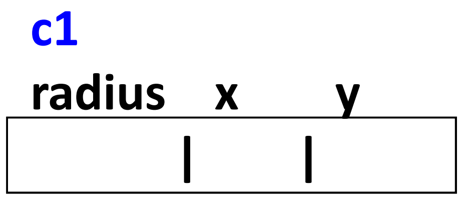
```
int contain(struct circle *c1, struct circle *c2)
```

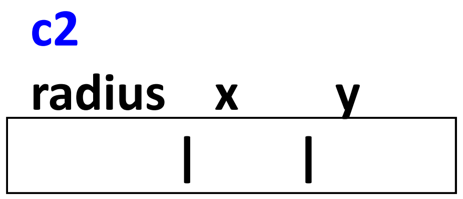
Sample input and output sessions:

```
(1)  Test Case 1
Enter circle 1 (radius x y):
10 5 5
Enter circle 2 (radius x y):
5 1 1
Circle intersection: 1
Circle contain: 0

(2) Test Case 2
Enter circle 1 (radius x y):
10 5 5
Enter circle 2 (radius x y):
1 1 1
Circle intersection: 1
Circle contain: 1
```

Structures – Q1 (computeCircle)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define INIT_VALUE -1000
struct circle {
   double radius;
   double x;
   double y;
};
int intersect(struct circle, struct circle);
int contain(struct circle *, struct circle *);
int main()
{
   struct circle c1, c2;
   int choice, result = INIT_VALUE;

   printf("Select one of the following options: \n");
   printf("1: intersect()\n");
   printf("2: contain()\n");
   printf("3: exit()\n");
   do {
      result=-1;
      printf("Enter your choice: \n");
      scanf("%d", &choice);
      switch (choice) {
```





Structures – Q1 (computeCircle)

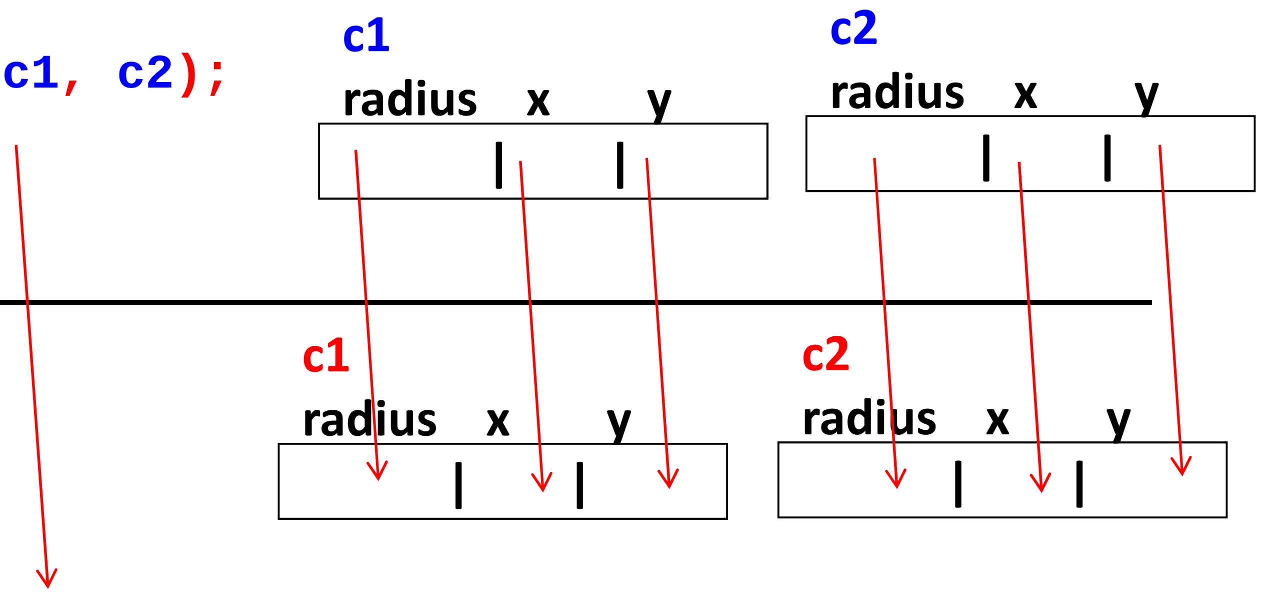
```
        case 1:
           printf("Enter circle 1 (radius x y): \n");
           scanf("%lf %lf %lf", &c1.radius, &c1.x, &c1.y);
           printf("Enter circle 2 (radius x y): \n");
           scanf("%lf %lf %lf", &c2.radius, &c2.x, &c2.y);
           result = intersect(c1, c2);
           if (result == 1)
              printf("intersect(): intersect\n");
           else if (result == 0)
              printf("intersect(): not intersect\n");
           else
              printf("intersect(): error\n");
           break;
        case 2:
           printf("Enter circle 1 (radius x y): \n");
           scanf("%lf %lf %lf", &c1.radius, &c1.x, &c1.y);
           printf("Enter circle 2 (radius x y): \n");
           scanf("%lf %lf %lf", &c2.radius, &c2.x, &c2.y);
           result = contain(&c1, &c2);
           if (result == 1)
              printf("contain(): contain\n");
           else if (result == 0)
              printf("contain(): not contain\n");
           else
              printf("contain(): error\n");
           break;
     }
   } while (choice < 3);
   return 0;
}
```

Use dot notation when accessing members of the structure.

Structures – Q1 (computeCircle)

```
int main()
{
  result = intersect(c1, c2);

}
```

Call by value



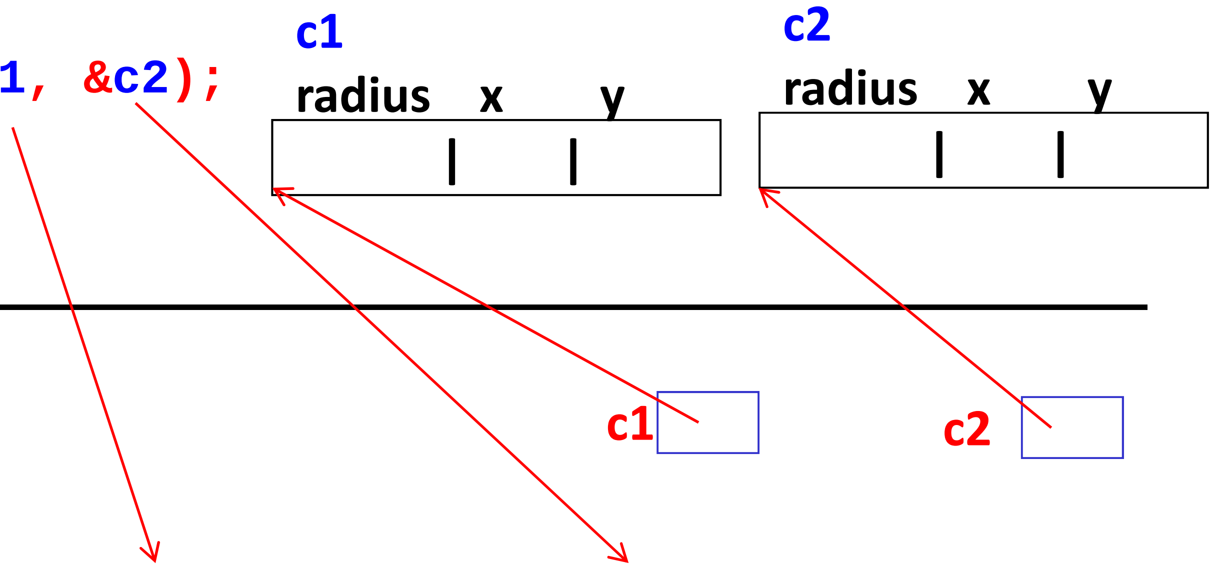
```
int intersect(struct circle c1, struct circle c2)
{
   double a, b;

   a = c1.x - c2.x;
   b = c1.y - c2.y;
   return (sqrt(a*a + b*b) <= (c1.radius + c2.radius));
}
```

Use dot notation when accessing members of the structure in this function.

Structures – Q1 (computeCircle)

```
int main()
{
  result = contain(&c1, &c2);


}
```

Call by reference



```
int contain(struct circle *c1, struct circle *c2)
{
  double a, b;

  a = c1->x - c2->x;
  b = c1->y - c2->y;
  return (c1->radius >=(c2->radius+sqrt(a*a+b*b)));
}
```

Use -> notation when accessing members of the structure in this function.

Structures – Q2 (computeAverage)

Assume the following structure is defined to represent a grade record of a student:

```
struct student{
    char name[20];          /* student name */
    double testScore;       /* test score */
    double examScore;       /* exam score */
    double total; /* total = (testScore+examScore)/2 */
};
```

Write a C program to create a database of maximum 50 students using an array of structures.

1. The program takes in the number of students in the class.
2. For each student, it takes in the test score and exam score. Then it computes and prints the **total score** for each student. The input will end when the student name is "**END**".
3. Then, it computes and prints the total score for each student, and computes and prints the **average score** of all students.

Sample input and output session:

```
Enter student name:
Hui Cheung
Enter test score:
34
Enter exam score:
46
Student Hui Cheung total = 40.00
Enter student name:
Tan May
Enter test score:
60
Enter exam score:
80
Student Tan May total = 70.00
Enter student name:
END
average(): 55.00
```

Structures – Q2 (computeAverage)

```
#include <stdio.h>
#include <string.h>
struct student{
   char name[20]; /* student name */
   double testScore; /* test score */
   double examScore; /* exam score */
   double total;  /* total = (testscore+examscore)/2 */
};
double average();
int main()
{
   printf("average(): %.2f\n", average());
   return 0;
}
```

Structures – Q2 (computeAverage)

```
double average(){
   struct student stud[50];
   double sum = 0;
   int i;
   char *p;
   i=0;
   printf("Enter student name: \n");
   fgets(stud[i].name, 80, stdin);
   if (p=strchr(stud[i].name,'\n')) *p = '\0';
   while (strcmp(stud[i].name, "END")!=0)          // use strcmp()
   {
      /* get scores */
      printf("Enter test score: \n");
      scanf("%lf", &stud[i].testScore);            // use dot notation
      printf("Enter exam score: \n");
      scanf("%lf", &stud[i].examScore);            // use dot notation
      /* compute total */
      stud[i].total = (stud[i].testScore + stud[i].examScore)/2;
      printf("Student %s total = %.2f\n", stud[i].name, stud[i].total);
      sum += stud[i].total;
      i++;
      printf("Enter student name: \n");
      scanf("\n");      // remove the remaining char in the buffer
      fgets(stud[i].name, 80, stdin);
      if (p=strchr(stud[i].name,'\n')) *p = '\0';
   }
   if (i != 0)
      return (sum/i);
   else
      return 0;
```

Structures – Q3 (book)

Write a program that processes an array of book records. For each book record, it stores the title of the book, author name (last name and first name), date of publication and publisher. In the program, it reads in each book's information, and then prints the book information on the display. The functions **readBook()** and **printBook()** are used for the reading and printing of each book record. The program should repeatedly read in book records from the user and print the book informtion on the screen until all book records are read. The prototypes of the two functions are given below:

```
void readBook(Booktype *book);
void printBook(Booktype book);
```

The structure definition for Booktype is given below:

```
typedef  struct {
  char title[81];
  char lastname[81];
  char firstname[81];
  char publisher[81];
  int day, month, year;
} Booktype;
```

Sample input and output session:

Test Case 1:

```
Enter the title of the book:
C Programming
Enter the author's first name:
Hui
Enter the author's last name:
SC
Enter the publisher name:
Pearson
The book information:
Title: C Programming
Author: Hui SC
Publisher: Pearson
```

Structures – Q3 (book)

```
#include <stdio.h>
#include <string.h>
typedef  struct {
  char title[80];
  char lastname[80];
  char firstname[80];
  char publisher[80];
} Booktype;
void readBook(Booktype *book);
void printBook(Booktype book);
int main()
{
  Booktype book;

  readBook(&book);
  printf("The book information: \n");
  printBook(book);
}
```

Structures – Q3 (book)

```
void readBook(Booktype *book)
{
  char *p;

  printf("Enter the title of the book: \n");
  fgets(book->title, 80, stdin);
  if (p=strchr(book->title,'\n')) *p = '\0';

  printf("Enter the author first name: \n");
  fgets(book->firstname, 80, stdin);
  if (p=strchr(book->firstname,'\n')) *p = '\0';

  printf("Enter the author last name: \n");
  fgets(book->lastname, 80, stdin);
  if (p=strchr(book->lastname,'\n')) *p = '\0';

  printf("Enter the publisher name: \n");
  fgets(book->publisher, 80, stdin);
  if (p=strchr(book->publisher,'\n')) *p = '\0';
}
void printBook(Booktype book)
{
  printf("Title: %s\n", book.title);
  printf("Author: %s %s\n", book.firstname, book.lastname);
  printf("Publisher: %s\n", book.publisher);
}
```

Structures – Q4 (mayTakeLeave)

Given the following information:

```
typedef struct {
        int id;         /* staff identifier */
        int totalLeave; /* the total number of days of leave
allowed */
        int leaveTaken; /* the number of days of leave taken so
far */
} leaveRecord;
```

write the code for the following functions:

(a) void getInput(leaveRecord list[], int *n);

Each line of the input has three integers representing one staff identifier, his/her total number of days of leave allowed and his/her number of days of leave taken so far respectively. The function will read the data into the array *list* until end of input and returns the number of records read through *n* .

(b) int mayTakeLeave(leaveRecord list[], int id, int leave, int n);

It returns 1 if a leave application for *leave* days is approved. Staff member with identifier *id* is applying for *leave* days of leave. *n* is the number of staff in *list*. Approval will be given if the leave taken so far plus the number of days applied for is less than or equal to his total number of leave days allowed. If approval is not given, it returns 0. It will return -1 if no one in *list* has identifier *id*.

(c) void printList(leaveRecord list[], int n);

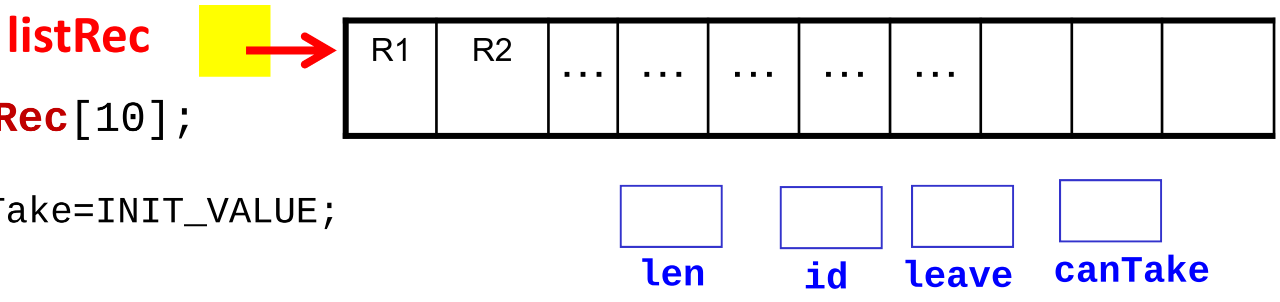
It prints the list of leave records of each staff. *n* is the number of

Sample input and output sessions:

```
(1) Test Case 1
Enter the number of staff records:
2
Enter id, totalleave, leavetaken:
11 28 25
Enter id, totalleave, leavetaken:
12 28 6
The staff list:
id = 11, totalleave = 28, leave taken = 25
id = 12, totalleave = 28, leave taken = 6
Please input id, leave to be taken:
11 6
The staff 11 cannot take leave
(2) Test Case 2
Enter the number of staff records:
2
Enter id, totalleave, leavetaken:
11 28 25
Enter id, totalleave, leavetaken:
12 28 6
The staff list:
id = 11, totalleave = 28, leave taken = 25
id = 12, totalleave = 28, leave taken = 6
Please input id, leave to be taken:
12 6
The staff 12 can take leave
```

Structures – Q4 (mayTakeLeave)

```
#include <stdio.h>
#define INIT_VALUE 1000
typedef struct {
   int id;             /* staff identifier */
   int totalLeave;     /* the total number of days of leave allowed */
   int leaveTaken;     /* the number of days of leave taken so far */
} leaveRecord;
int mayTakeLeave(leaveRecord list[], int id, int leave, int n);
void getInput(leaveRecord list[], int *n);
void printList(leaveRecord list[], int n);

int main()
{
   leaveRecord listRec[10];
   int len;
   int id, leave, canTake=INIT_VALUE;
   int choice;

   printf("Select one of the following options: \n");
   printf("1: getInput()\n");
   printf("2: printList()\n");
   printf("3: mayTakeLeave()\n");
   printf("4: exit()\n");
   do {
      printf("Enter your choice: \n");
      scanf("%d", &choice);
      switch (choice) {
```

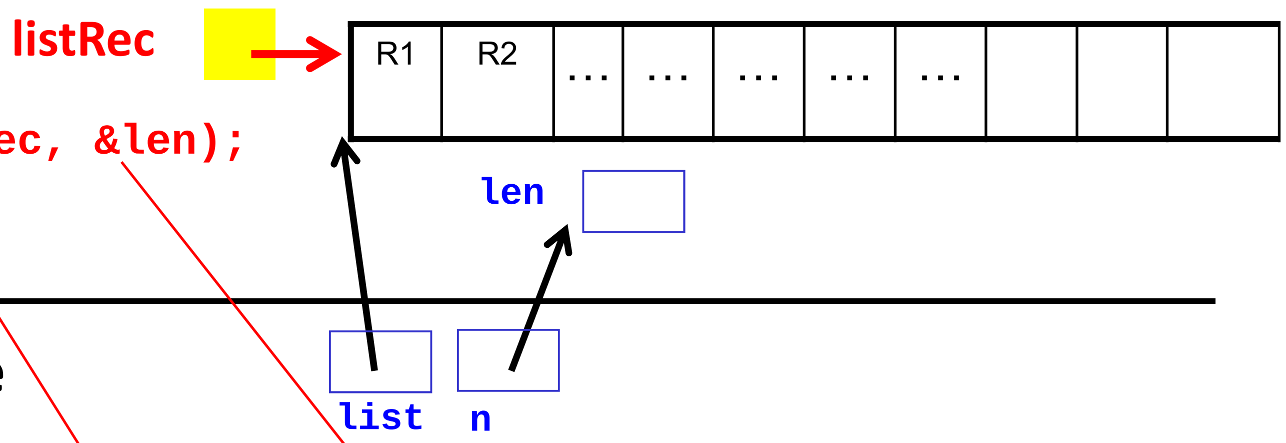


Structures – Q4 (mayTakeLeave)

```
        case 1:
           getInput(listRec, &len);
           printList(listRec, len);
           break;
        case 2:
           printList(listRec, len);
           break;
        case 3:
           printf("Please input id, leave to be taken: \n");
           scanf("%d %d", &id, &leave);
           canTake = mayTakeLeave(listRec, id, leave, len);
           if (canTake == 1)
              printf("The staff %d can take leave\n", id);
           else if (canTake == 0)
              printf("The staff %d cannot take leave\n", id);
           else if (canTake == -1)
              printf("The staff %d is not in the list\n", id);
           else
              printf("Error!");
           break;
     }
  } while (choice < 4);
  return 0;
}
```

Structures – Q4 (mayTakeLeave)

```
int main()
{

      getInput(listRec, &len);

}
```

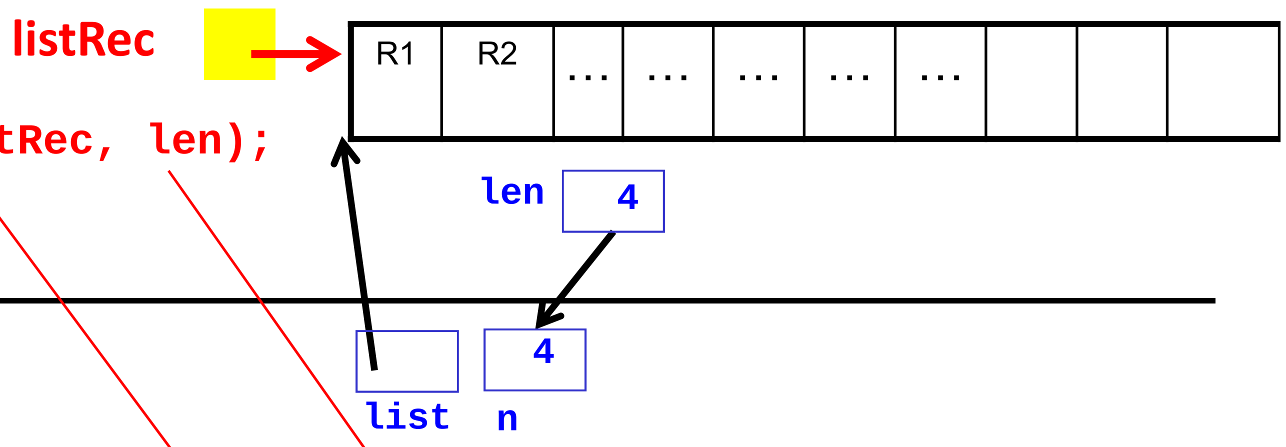


Call by reference

```
void getInput(leaveRecord list[], int *n)  // n – using call by reference
{
   int total;

   *n = 0;
   printf("Enter the number of staff records: \n");
   scanf("%d", &total);

   while ((*n) != total) {
      printf("Enter id, totalleave, leavetaken: \n");
      scanf("%d %d %d", &list[*n].id, &list[*n].totalLeave,
                        &list[*n].leaveTaken);
      (*n)++;
   }
}
```

Structures – Q4 (mayTakeLeave)

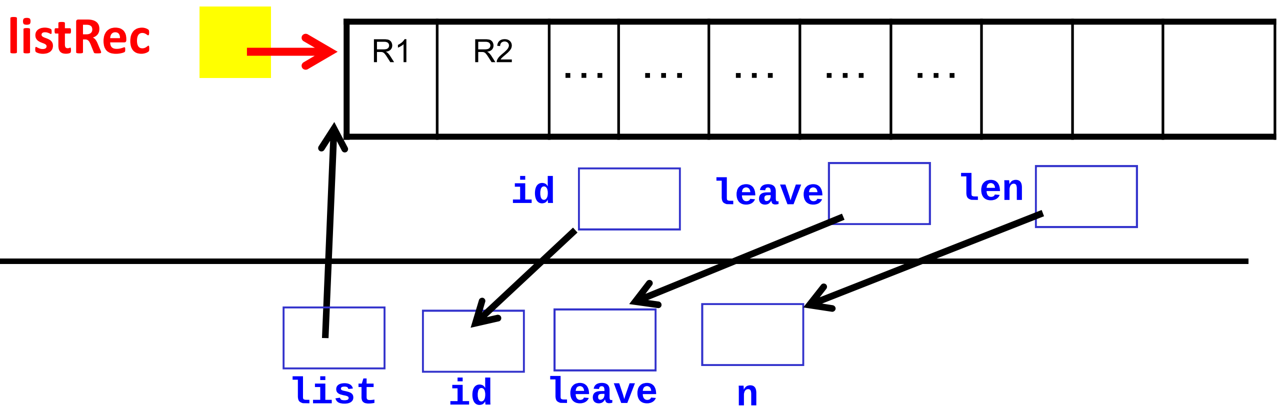
```
int main()
{

      printList(listRec, len);

}
```



```
void printList(leaveRecord list[], int n)
{
   int p;

   printf("The staff list:\n");
   for (p = 0; p < n; p++)
      printf ("id = %d, totalleave = %d, leave taken = %d\n",
         list[p].id, list[p].totalLeave, list[p].leaveTaken);
}
```

Structures – Q4 (mayTakeLeave)

```
int main()
{
 canTake = mayTakeLeave(listRec, id, leave, len);

}
```



```
 int mayTakeLeave(leaveRecord list[], int id, int leave, int n)
 {
    int p;

    for (p = 0; p < n; p++)
       if (list[p].id == id)
          return (list[p].totalLeave >= (list[p].leaveTaken + leave));

    return -1;
 }
```