System Programming (CSE4009)

Assignment #1 — Hotel Management System
(Due: Oct. 28, midnight)

Guest File (guests)

```
ganga01.cse.psu.edu 330% cat guests
keith haviland
dina gray
tom brady

jammie benn
tim cook

lebron james

stephen curry
```

- Your will build a hotel management system
- You will maintain a file named “guests” (on the left)
  - Each line represents a room (e.g., the 5th line indicates the room 5)
  - The string appeared in a line indicates the name of the guest in the corresponding room

Management System (frontdesk.c)

```
/*file for currnet guest information*/
#define FILE_NAME "guests"
/*max length of the name for a guest*/
#define NAME_LEN_MAX 21
/*total number of rooms */
#define NUM_ROOMS 10
```

```
ganga01.cse.psu.edu 330% cat guests
keith haviland
dina gray
tom brady

jammie benn
tim cook

lebron james

stephen curry
```

- A skeleton code “frontdesk.c” is given
  - FILE_NAME: you can change this
  - NAME_LEN_MAX: the size of each line is fixed to 21 bytes (21st byte is “\n”); do not change this
  - NUM_ROOMS: you can change this

Hotel Management (2)

```
if((fd = open(FILE_NAME, O_RDONLY)) == -1)
{
        printf("fail to open file\n");  exit(0);
}

for(i=1; i<=NUM_ROOMS; i++)
{

        offset = (i - 1) * NAME_LEN_MAX;

        if(lseek(fd, offset, SEEK_SET) == -1)
        {
                printf("faile to seek position\n"); exit(0);
        }

        nread = read(fd, namebuf, NAME_LEN_MAX);
        namebuf[NAME_LEN_MAX] = '\0';

        if(nread > 0)
        {
                printf("room # %d: %s", i, namebuf);
        }
        else
        {
                printf("room # %d:", i);
        }

        memset(namebuf, 0, NAME_LEN_MAX);
}

close(fd);
```

```
ganga01.cse.psu.edu 331% ./exe
room # 1: keith haviland
room # 2: dina gray
room # 3: tom brady
room # 4:
room # 5: jammie benn
room # 6: tim cook
room # 7:
room # 8: lebron james
room # 9:
room # 10: stephen curry
```

- print_guests() is defined
  - Open the file
  - Read each line by changing offset
  - Examine the contents of each line

Define Function (1) - check_vacancies()

```c
/* hw#1-1 define this function
 *this fuction finds the number of empty rooms
 *return  "the SMALLEST ROOM NUMBER among empty rooms" if successful & and print out "the number of empty rooms is XXX";
 *return 0 if there are no available rooms & print out "there are no available rooms"
 */
int check_vacancies()
{
        int empty_room = 0;
        int smallest = -1;

        //add your code here

        if (empty_room)
        {
                printf("the number of empty rooms is %d\n",empty_room);
                return smallest;
        }
        else
        {
                printf("there are no available rooms\n");
                return 0;
        }
}
```

- You are required to define a function named — int check_vacancies()
- This function finds the total number of empty rooms
- If there are some available rooms
  - Print out “the number of empty rooms is XXX”
  - Return the SMALLEST room number among empty rooms
- If there are no available rooms
  - Print out “there are no available rooms”
  - Return 0

Define Function (2) - checkin_guest()

```
/* hw#1-2 define this function
 *1st arg: guest name
 *2nd arg: room # to assign
 *this fuction adds guest name (1st arg) to the corresponding file position (2nd arg)
 *return  0 if successful;
 *return -1 if the room is occupied & print out "room # <2nd arg> is occupied by XXX"
 *return -1 if the room number exceeds NUM_ROOMS & print out "room # <2nd arg> does not exist"
 */
int checkin_guest(char *guestname, int roomnum)
{
        //add your code here

        return -1;
}
```

- You are required to define a function named — int checkin_guest(char* name, int room)
- This function adds name (1st arg) to the location of room (2nd arg)
- If the addition is successful, return 0
- If the requested room is occupied, return -1
  - Print out “room # <2nd arg> is occupied by XXX”
- If the requested room # is larger than NUM_ROOMS, return -1
  - Print out “room # <2nd arg> does not exist”

Define Function (3) - checkout_guest()

```
/* hw#1-3 define this function
 *1st arg: guest name
 *this fuction eliminates guest name (1st arg) from the list
 *return "room #" that beomces empty if successful;
 *return -1 if the given name (1st arg) is not in the list & print out "guest XXX does not exist in the room"
 */
int checkout_guest(char *guestname)
{
        //add your code here


        printf("guest %s does not exist in the room\n",guestname);
        return -1;
}
```

- You are required to define a function named — int checkout_guest(char* name)
- This function finds where the name is and eliminates it from the list
- If the deletion is successful, return his/her “room #”
- If the given name is not in the list, return -1
  - Print out “guest <1st arg> does not exist in the rooms”

Submission and Other Details

- Skeleton is given
  - Add your code to “frontdesk.c”
  - A sample list (“guests”) is given (NUM_ROOMS = 10)
- Submission
  - Due: Oct 28 (Fri) at midnight
  - Rename your source file to “frontdesk_name_id.c”
  - Upload your source file to LMS by the due
- Evaluation
  - Using various test cases, the three functions will be evaluated
  - If any cheating is detected, both provider and cheater will get an F
- Important notice
  - Try to keep the size of each line unchanged - NAME_LEN_MAX (20 bytes + “\n”)
  - When you change a line, you should write 21 bytes
  - This is because other functions assume that each line take 21 bytes