1번

- 코드

```c
#include <stdlib.h>
#include <fcntl.h>
#include <stdio.h>

char *workfile="junk";

int main(){
        int filedes;

        /* "junk" 파일이 없는 경우 */
        if ((filedes = open (workfile, O_RDWR)) == -1){
                printf ("Couldn't open %s\n", workfile);
                exit(1);
        }

        exit(0);
}
~
```

- 실행 결과1(파일이 존재할 때)

```
boo@boo-VirtualBox:~/UNIX$ gcc p1.c -o p1
boo@boo-VirtualBox:~/UNIX$ ./p1
junk is exist
```

- 실행 결과2(파일이 존재하지 않을 때)

```
boo@boo-VirtualBox:~/UNIX$ ./p1
Couldn't open junk
```

3번

- 코드

```c
#include <stdlib.h>
#include <fcntl.h>
#include <stdio.h>

#define PERMS 0644

char *filename= "newfile";

int main(){
        int filedes; /* 파일 디스크립터 */

        /* 파일 생성 */
        if ((filedes = creat("./newfile", PERMS)) == -1){
                /* 에러가 있는 경우 */
                printf("Couldn't create %s\n", filename);
                exit(1);
        }
        else{
                printf("%s is created\n", filename);
        }

        /* 파일 open */
        if ((filedes = open (filename, O_RDWR, PERMS)) == -1){
                /* 에러가 있는 경우 */
                printf("Couldn't open %s\n", filename);
                exit(1);
        }
        else {
                printf("%s is opened\n", filename);
        }

        exit(0);
}
```

- 실행 결과

```
boo@boo-VirtualBox:~/UNIX$ gcc p3.c -o p3
boo@boo-VirtualBox:~/UNIX$ ./p3
newfile is created
newfile is opened
```

5번

- 코드

```c
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>

#define BUFSIZE 512
```

```c
int main(int argc, char *argv[]){
        /* 명령이 잘못된 경우 */
        if(argc != 2) {
                printf("Wrong input. Usage : %s <file name>\n", argv[0]);
                exit(1);
        }

        char buffer[BUFSIZE];  /* 문자열 저장 */
        int filedes;  /* 파일 디스크립터 */
        int cnt_word = 0;  /* 단어 개수 */
        int cnt_line = 0;  /* 줄의 수 */

        ssize_t nread; /* 읽은 크기 */

        char *filename= argv[1]; /* 파일 이름 */

        /* 파일 오픈 */
        if (( filedes = open(filename, O_RDONLY)) == -1){
                printf("error in opening %s\n", filename);
                exit(1);
        }

        /* 파일 읽기 */
        while( (nread = read(filedes, buffer, BUFSIZE)) > 0){
                 /* 단어, 줄의 개수 탐색 */
                for (int i = 0; i < nread; i++){
                        if( buffer[i] == '\n' || buffer[i] == ' ' || buffer[i] == '\t'){
                                cnt_word++;
                        }
                        if(buffer[i] == '\n'){
                                cnt_line++;
                        }
                }
        }

        printf("number of words: %d\n", --cnt_word);
        printf("number of lines: %d\n", --cnt_line);
        exit(0);

}
```

- Example 파일

```
This is example file for exercise05.
This is example file for exercise05
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"example" 2L, 73C                                          2,35          All
```

- 실행 결과

```
boo@boo-VirtualBox:~/Downloads/unix/UNIX_hw-master$ gcc p5.c -o p5
boo@boo-VirtualBox:~/Downloads/unix/UNIX_hw-master$ ./p5 example
number of words: 12
number of lines: 2
```

7번

- 코드

```c
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>

#define BUFFSIZE 512
#define PERM 0644

/* 파일 복사 함수 */
/* name1: 복사 파일 이름, name2: 붙여넣기 파일 이름 */
int mycp(const char *name1, const char *name2){
        int infile, outfile;  /* 복사 파일, 붙여넣기 파일 디스크립터 */
        ssize_t nread;  /* 읽은 크기 */
        char buffer[BUFFSIZE];  /* 문자열 저장 */

        /* 복사 파일 오픈 */
        if ( (infile = open (name1, O_RDONLY)) == -1){
                return (-1);
        }

        /* 붙여넣기 파일 오픈 */
        if ( (outfile = open (name2, O_WRONLY|O_CREAT|O_TRUNC, PERM)) == -1){
                close(infile);
                return (-2);
        }

        /* 파일 내용 복사 */
        while( (nread = read (infile, buffer, BUFFSIZE) ) > 0 ){
                if (write(outfile, buffer, nread) < nread){
                        close(infile);
                        close(outfile);
                        return (-3);
                }
        }

        /* 파일 닫기 */
        close(infile);
        close(outfile);

        if (nread == -1){
                return (-4);
        }
        else{
                return (0);
        }
}
```

```c
int main(int argc, char *argv[]){
        /* 명령이 잘못된 경우 */
        if(argc != 3) {
                printf("Wrong input. Usage : %s <file name1> <file name2>\n", argv[0]);
                return 0;
        }

        char *filename1 = argv[1];
        char *filename2 = argv[2];

        mycp(filename1, filename2);
}
```

- File1

```
This is file1
~
~
~
~
```

- 실행 결과와 그에 따라 생성된 file2

```
boo@boo-VirtualBox:~/UNIX$ gcc p7.c -o p7
boo@boo-VirtualBox:~/UNIX$ ./p7 file1 file2
boo@boo-VirtualBox:~/UNIX$
```

```
This is file1
~
~
~
~
~
~
```

9번

- 코드

```c
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>

#define NAMELENGTH 41
#define NROOMS 10

char namebuf[NAMELENGTH];  /* 문자열 저장 */
int infile = -1;  /* 파일 디스크립터 */
off_t offset = 0;  /* 파일 오프셋 */
int freeroom = NROOMS;  /* 빈 방 중 방 번호가 가장 작은 방 */
```

```c
/* 방 조사 함수 */
/* roomno: 방 번호 */
/* 방이 빈 경우 empty, 아닌 경우 거주자 이름 리턴 */
char *getoccupier(int roomno){
        ssize_t nread;

        /* file open */
        if( infile == -1 && (infile = open("residents", O_RDONLY)) == -1){
                return (NULL);
        }

        /* 방 위치 찾은 후, 투숙객 이름 읽기 */
        if (lseek(infile, offset, SEEK_SET) == -1){
                return (NULL);
        }

        /* read file */
        if( (nread = read(infile, namebuf, NAMELENGTH) ) <= 0){
                return (NULL);
        }

        /* 방이 비어있을 경우 */
        if(namebuf[0] == ' ' && namebuf[1] == ' '){
                /* findfree를 위해 가장 방 번호가 작은 빈방인지 확인 */
                if(roomno < freeroom){
                        freeroom = roomno;
                }
                offset += NAMELENGTH;
                char *empty = "Empty";
                return(empty);
        }

        /* 거주자가 있을 경우 */

        offset += NAMELENGTH;

        /* 개행문자를 마지막에 넣어 문자열 생성 */
        namebuf[NAMELENGTH - 1] = '\0';
        return (namebuf);
}
```

```c
/* 빈 방 탐색 함수 */
/* 빈 방 중 방 번호가 가장 작은 방 리턴 */
int findfree(){
        offset = 0;
        for(int j = 1; j<= NROOMS; j++){
                getoccupier(j);
        }
        return freeroom;
}

int main(){
        int j;
        char *getoccupier (int), *p;

        printf("========== getoccupier ==============\n");
        offset = 0;
        for(j = 1; j<= NROOMS; j++){
                if(p = getoccupier(j)){
                        printf("room %2d, %s\n", j, p);
                } else{
                        printf("Error on room %2d\n", j);
                }
        }

        printf("============= findfree ===============\n");
        printf("free roome is %2d\n", findfree());
}
```

- 실행결과

```
========== getoccupier ==============
room  1, Kim
room  2, Lee
room  3, Empty
room  4, Empty
room  5, Bixby
room  6, Choi
room  7, Empty
room  8, Empty
room  9, Empty
room 10, Empty
============= findfree ===============
free roome is  3
```

10번

- 코드

```c
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>

#define NAMELENGTH 41
#define NROOMS 10

char namebuf[NAMELENGTH];  /* 문자열 저장 */
int infile = -1;  /* 파일 디스크립터 */
off_t offset = 0;  /* 파일 오프셋 */
int minFreeRoom = NROOMS;  /* 빈 방 중 방 번호가 가장 작은 방 */
```

```c
/* 방을 비우는 함수 */
/* rn: 비우고자 하는 방 번호 */
int freeroom(int rn){

        offset = NAMELENGTH * (rn - 1); /* 방 위치 offset */
        /* 이름을 지우기 위한 문자열 */
        char* blank = "                                        \n";

        /* file open */
        if( infile == -1 && (infile = open("residents", O_WRONLY )) == -1){
                return (-1);
        }

        /* 방 위치 찾기 */
        if (lseek(infile, offset, SEEK_SET) == -1){
                return (-1);
        }

        /* 이름 지우기 */
        if (write( infile, blank, NAMELENGTH - 1) == -1)
                return (-1);

        return 1;
}
```

```c
/* 빈 방인지 조사하고 방을 채우는 함수 */
/* rn: 방 번호, name: 게스트 이름 */
int addguest(int rn, char* origname){
        ssize_t nread;  /* 읽은 바이트 수 */
        /* 방 위치 offset */
        offset = NAMELENGTH * (rn - 1);
        char name[41] = { ' ' };
        name[40] = '\n';

        for(int i = 0; i < NAMELENGTH; i++){
                if(origname[i] == '\0')
                        break;
                name[i] = origname[i];
        }

        /* file open */
        if( infile == -1 && (infile = open("residents", O_RDWR)) == -1){
                return (-1);
        }

        /* 방 위치 찾기*/
        if (lseek(infile, offset, SEEK_SET) == -1){
                return (-1);
        }

        /* read file */
        if( (nread = read(infile, namebuf, NAMELENGTH) ) <= 0){
                return (-1);
        }

        /* 방이 비어있지 않을 경우 */
        if(namebuf[0] != ' ' && namebuf[1] != ' '){
                return 0;
        }

        /* 이름 추가하기 */
        if (lseek(infile, offset, SEEK_SET) == -1){
                return (-1);
        }

        if (write( infile, name, NAMELENGTH) == -1)
                return (-1);

        return 1;
}
```

```c
int main(){
        /* 명령 입력 받기 */
        printf("1: freeroom\n");
        printf("2: addguest\n");
        printf("Input your command number: ");

        char x = getchar();

        /* 명령 실행 */
        if(x == '1') {
                printf("========== freeroom ==============\n");
                printf("Input room number to free: ");
                char y[2];
                scanf("%s", y);
                int rn = atoi(y);
                if(freeroom(rn))
                        printf("room %2d is free\n", rn);
                else
                        printf("Error on freeroom\n");
        } else if (x == '2'){
                printf("========== addguest ==============\n");
                printf("Input room number to add guest: ");
                char y[2];
                scanf("%s", y);
                int rn = atoi(y);

                printf("Input guest name: ");
                char name[41];
                scanf("%s", name);
                name[40] = '\n';
                name[41] = '\0';

                int res;
                if( res = addguest(rn, name))
                        printf("Guest is added to room %2d\n", rn);
                else if(res == 0)
                        printf("Room %2d is not empty\n", rn);
                else
                        printf("Error on addguest\n");
        } else{
                printf("Wrong input\n");
        }
}
```

- 실행결과

```
1: freeroom
2: addguest
Input your command number: 1
========== freeroom ==============
Input room number to free: 1
room  1 is free
```

11번

- 코드

```c
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>

#define NAMELENGTH 41
#define NROOMS 10

char namebuf[NAMELENGTH];  /* 문자열 저장 */
int infile = -1;  /* 파일 디스크립터 */
off_t offset = 0;  /* 파일 오프셋 */
int minFreeRoom = NROOMS;  /* 빈 방 중 방 번호가 가장 작은 방 */
```

```c
/* 방 조사 함수 */
/* roomno: 방 번호 */
/* 방이 빈 경우 empty, 아닌 경우 거주자 이름 리턴 */
char *getoccupier(int roomno){
        ssize_t nread;

        /* file open */
        if( infile == -1 && (infile = open("residents", O_RDONLY)) == -1){
                return (NULL);
        }

        /* 방 위치 찾은 후, 투숙객 이름 읽기 */
        if (lseek(infile, offset, SEEK_SET) == -1){
                return (NULL);
        }

        /* read file */
        if( (nread = read(infile, namebuf, NAMELENGTH) ) <= 0){
                return (NULL);
        }

        /* 방이 비어있을 경우 */
        if(namebuf[1] == ' '){
                /* findfree를 위해 가장 방 번호가 작은 빈방인지 확인 */
                if(roomno < minFreeRoom){
                        minFreeRoom = roomno;
                }
                offset += NAMELENGTH;
                char *empty = "Empty";
                return(empty);
        }

        /* 거주자가 있을 경우 */

        offset += NAMELENGTH;

        /* 개행문자를 마지막에 넣어 문자열 생성 */
        namebuf[NAMELENGTH - 1] = '\0';
        return (namebuf);
}
```

```c
/* 빈 방 탐색 함수 */
/* 빈 방 중 방 번호가 가장 작은 방 리턴 */
int findfree(){
        offset = 0;
        minFreeRoom = NROOMS;
        for(int j = 1; j<= NROOMS; j++){
                getoccupier(j);
        }
        return minFreeRoom;
}

/* 방을 비우는 함수 */
/* rn: 비우고자 하는 방 번호 */
int freeroom(int rn){

        offset = NAMELENGTH * (rn - 1); /* 방 위치 offset */
        /* 이름을 지우기 위한 문자열 */
        char* blank = "                                                \n";

        /* file open */
        if( infile == -1 && (infile = open("residents", O_WRONLY )) == -1){
                return (-1);
        }

        /* 방 위치 찾기 */
        if (lseek(infile, offset, SEEK_SET) == -1){
                return (-1);
        }

        /* 이름 지우기 */
        if (write( infile, blank, NAMELENGTH - 1) == -1)
                return (-1);

        return 1;
}
```

```c
/* 빈 방인지 조사하고 방을 채우는 함수 */
/* rn: 방 번호, name: 게스트 이름 */
int addguest(int rn, char* orignname){
        ssize_t nread;  /* 읽은 바이트 수 */
        /* 방 위치 offset */
        offset = NAMELENGTH * (rn - 1);
        char name[41] = { ' ' };
        name[40] = '\n';

        for(int i = 0; i < NAMELENGTH; i++){
                if(orignname[i] == '\0')
                        break;
                name[i] = orignname[i];
        }

        /* file open */
        if( infile == -1 && (infile = open("residents", O_RDWR)) == -1){
                return (-1);
        }

        /* 방 위치 찾기*/
        if (lseek(infile, offset, SEEK_SET) == -1){
                return (-1);
        }

        /* read file */
        if( (nread = read(infile, namebuf, NAMELENGTH) ) <= 0){
                return (-1);
        }

        /* 방이 비어있지 않을 경우 */
        if(namebuf[0] != ' ' && namebuf[1] != ' '){
                return 0;
        }

        /* 이름 추가하기 */
        if (lseek(infile, offset, SEEK_SET) == -1){
                return (-1);
        }

        if (write( infile, name, NAMELENGTH) == -1)
                return (-1);

        return 1;
}
```

```c
int main(){
        /* 명령 입력 받기 */
        printf("========== frontdesk ==========\n");
        printf("1: getoccupier\n");
        printf("2: freeroom\n");
        printf("3: addguest\n");
        printf("4: findfree\n");
        printf("Input your command number: ");

        char x = getchar();

        /* 명령 실행 */
        if(x == '1') {
                int j;
                char *getoccupier (int), *p;

                printf("========== getoccupier ==============\n");
                offset = 0;
                for(j = 1; j<= NROOMS; j++){
                        if(p = getoccupier(j)){
                                printf("room %2d, %s\n", j, p);
                        } else{
                                printf("Error on room %2d\n", j);
                        }
                }
        } else if(x == '2'){
                printf("========== freeroom ==============\n");
                printf("Input room number to free: ");
                char y[2];
                scanf("%s", y);
                int rn = atoi(y);
                if(freeroom(rn))
                        printf("room %2d is free\n", rn);
                else
                        printf("Error on freeroom\n");
        } else if(x == '3'){
                printf("========== addguest ==============\n");
                printf("Input room number to add guest: ");
                char y[2];
                scanf("%s", y);
                int rn = atoi(y);
```

```
                        printf("Error on addguest\n");
        } else {
                printf("============= findfree ===============\n");
                int res = findfree();
                if(res)
                        printf("free roome is %2d\n", findfree());
                else
                        printf("Error on findfree\n");
        }
}
```

- 실행결과1: getoccupier

```
========== frontdesk ==========
1: getoccupier
2: freeroom
3: addguest
4: findfree
Input your command number: 1
========== getoccupier ==============
room  1, Kim
room  2, Lee
room  3, Empty
room  4, Serna
room  5, Bixby
room  6, Empty
room  7, Empty
room  8, Empty
room  9, Empty
room 10, Empty
```

- 실행결과 2: freeroom과 그에따른 getoccupier 결과

```
========== frontdesk ==========
1: getoccupier
2: freeroom
3: addguest
4: findfree
Input your command number: 2
========== freeroom ==============
Input room number to free: 4
room  4 is free
```

```
========== frontdesk ==========
1: getoccupier
2: freeroom
3: addguest
4: findfree
Input your command number: 1
========== getoccupier ==============
room  1, Kim
room  2, Lee
room  3, Empty
room  4, Empty
room  5, Bixby
room  6, Empty
room  7, Empty
room  8, Empty
room  9, Empty
room 10, Empty
```

- 실행결과3: addguest와 그에따른 getoccupier 결과

```
========== frontdesk ==========
1: getoccupier
2: freeroom
3: addguest
4: findfree
Input your command number: 3
========== addguest ==============
Input room number to add guest: 6
Input guest name: Choi
Guest is added to room  6
```

```
========== frontdesk ==========
1: getoccupier
2: freeroom
3: addguest
4: findfree
Input your command number: 1
========== getoccupier ==============
room  1, Kim
room  2, Lee
room  3, Empty
room  4, Empty
room  5, Bixby
room  6, Choi
room  7, Empty
room  8, Empty
room  9, Empty
room 10, Empty
```

- 실행결과4: findfree

```
========== frontdesk ==========
1: getoccupier
2: freeroom
3: addguest
4: findfree
Input your command number: 4
============= findfree ===============
free roome is  3
```

14번

- 코드

```c
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>

#define NAMELENGTH 42
#define NROOMS 10
#define BUFFSIZE 512
#define PERM 0644


int filedes = -1; // 파일 디스크립터


int fileopen(char* filename, int flag){
        /* file open */
        if ((filedes = open (filename, flag)) == -1){
                printf ("Couldn't open %s\n", filename);
                return -1;
        }
        return filedes;
}
```

```c
int main(int argc, char *argv[]){
        /* 명령이 잘못된 경우 */
        if(argc != 3) {
                printf("Wrong input. Usage : %s <file name1> <open option>\n", argv[0]);
                return -1;
        }

        int res = 0;

        /* 일기와 쓰기용으로 파일을 개방 */
        if ( argv[2][0] == 'r' && argv[2][1] == 'w'){
                if ( (res = fileopen (argv[1], 2)) == -1){
                        close(filedes);
                        return (-1);
                } else{
                        printf("The file is opened with write and read mode\n");
                }
        }
        /* 읽기 전용으로 파일 개방 */
        else if (argv[2][0] == 'r'){
                if ( (res = fileopen (argv[1], 0)) == -1){
                        close(filedes);
                        return (-1);
                } else{
                        printf("The file is opened with read-only\n");
                }
        }
        /* 쓰기 전용으로 파일 개방 */
        else if( argv[2][0] == 'w'){
                if ( (res = fileopen (argv[1], 1)) == -1){
                        close(filedes);
                        return (-1);
                } else{
                        printf("The file is opened with write-only\n");
                }
        }
        /* 자료를 화일의 끝에 덧붙이기 위해 화일을 개방 */
        else if (argv[2][0] == 'a') {
                if ( (res = fileopen (argv[1], O_WRONLY | O_APPEND)) == -1){
                        close(filedes);
                        return (-1);
                } else{
                        printf("The file is opened with append mode\n");
                }
        } else {
                printf("Wrong Input\n");

        }

        printf("filedes: %d\n", res);

        return res;
}
```

- 실행 결과1: 읽기 전용으로 오픈했을 때

```
boo@boo-VirtualBox:~/Downloads$ ./excercise14 sample r
The file is opened with read-only
filedes: 3
```

15번

- 코드

```c
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>

#define SIZE 512

size_t nread;  // 읽은 파일 내용 바이트수
char buf[SIZE];  // 문자열 저장
int filedes = -1; // 파일 디스크립터
```

```c
/* 파일을 열고 내용을 표준 출력으로 복사하는 함수 */
int fileread(char *filename){
        printf("========== %s ==========\n", filename);

        /* file open */
        if ((filedes = open (filename, O_RDONLY)) == -1){
                printf ("Couldn't open %s\n", filename);
                return -1;
        }

        /* file read */
        while( (nread = read(filedes, buf, SIZE) ) > 0 ){
                write(1, buf, nread);
        }

        return 1;
}
```

```c
int main(int argc, char *argv[]){
        /* 인수가 없는 경우 */
        if(argc == 1) {
                char filename[SIZE];
                /* 파일 이름을 표준 입력으로 받기 */
                printf("Input filename\n");
                scanf("%s", filename);
                /* 표준 입력으로 받은 파일을 표준 출력으로 복사 */
                fileread(filename);
        }

        /* 인수가 있는 경우 */
        else{
                /* 인수로 받은 파일들의 내용을 표준 출력으로 복사 */
                for(int i = 1; i < argc; i++){
                        fileread(argv[i]);
                }
        }

        exit(0);
}
```

- sample1 파일, sample2 파일

```
This file is sample1
This file is sample1
~
```

```
This file is sample2
This file is sample2
```

- 실행결과1: 인수가 있을 때

```
boo@boo-VirtualBox:~/Downloads$ ./io sample1 sample2
========== sample1 ==========
This file is sample1
This file is sample1
========== sample2 ==========
This file is sample2
This file is sample2
```

- 실행경과2: 인수가 없을 때

```
boo@boo-VirtualBox:~/Downloads$ ./io
Input filename
sample1
========== sample1 ==========
This file is sample1
This file is sample1
```

실행결과3: 자신이 열 수 없는 파일이 있을 때

```
boo@boo-VirtualBox:~/Downloads$ ./io
Input filename
errorfile
========== errorfile ==========
Couldn't open errorfile
```