강의명: 프로그래밍
실습 번호: 7
실습 제목: structures (구조체)
학생 이름: 임지빈
학번: 202211051

1. 간단한 구조체
1.1
```
void init_student(STUDENT * s, char *name,
int number, char major)
{
 s-> name = name;
 s-> number = number;
 s-> major = major;
}

void print_student(STUDENT * s)
{
  printf("name=%s,number=%d,major=%d\n",s-
>name, s->number, s->major);
}
```
1.2

1.3
```
s2211051@oak:lab07$ gcc struct-simple.c -o struct-simple
s2211051@oak:lab07$ ./struct-simple
name=Chulsoo Kim,number=1000,major=77
name=Younghee Lee,number=2000,major=77
name=Christina Park,number=3000,major=88
```
2. 구조체의 배열
2.1

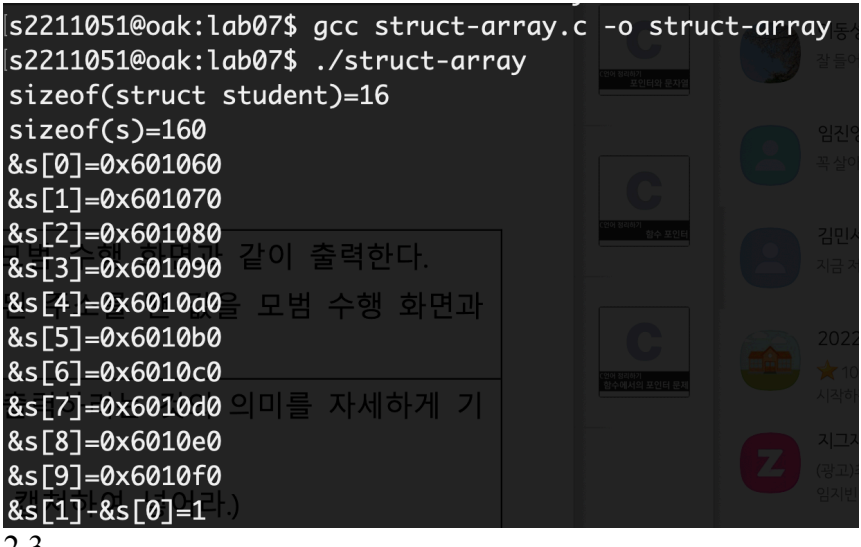
```
int main(void)
{
 int i;
 printf("sizeof(struct student)=%lu\n",
        sizeof (struct student));
 printf("sizeof(s)=%lu\n", sizeof(s));

 for (i=0; i<10; i++) {
   printf("&s[%d]=%p\n", i, &s[i]);
 }

 printf("&s[1]-&s[0]=%ld\n", &s[1]-&s[0]);
 return EXIT_SUCCESS;
}
```

구조체 student의 크기를 출력하려고 한다.
변수 s의 크기를 출력하려고 한다.
I(인덱스)가 0부터 9까지의 크기를 출력한다

2.2



2.3

3. 구조체와 malloc

3.1

```
STUDENT *alloc_student(int size)
{
STUDENT *alloc_student = malloc(size);
}
void init_student(STUDENT * s, char *name,
int number, char major)
{
 s-> name = name;
 s-> number = number;
 s-> major = major;
}

void print_student(STUDENT * s)
{
  printf("name=%s,number=%d,major=%d\n",s-
>name, s->number, s->major);
}
void link_student(STUDENT * s1, STUDENT *
s2)
{
s1->link=s2;
}
void free_student(STUDENT * s)
{
 free(s);
}
void print_all_student(STUDENT * s)
{
 print_student(s);
 print_student(s->link);
 print_student(s->link->link);
}
```

메인 함수에서 argument로 s1을 넣어주기 때문에 s가 s1으로 바뀌게 됨

3.2

Stack은 높은 주소쪽에 메모리가 저장되고, 데이터는 낮은쪽, 그리고 heap은 중간에 저장된다.

3.3

```
s2211051@oak:lab07$ gcc struct-malloc.c -o struct-malloc
s2211051@oak:lab07$./struct-malloc
name=Chulsoo Kim,number=1000,major=77
name=Younghee Lee,number=2000,major=77
name=Christina Park,number=3000,major=88
```

4. 메모리 영역들

4.1

```
void fun1(void)
{
 printf("&static_i=%p\n",&static_i);
 printf("&static_j=%p\n",&static_j);
}
```

주소를 출력하기 위해 &static_i를 출력하였다.
포인터이기 때문에 %p를 써줬다.

```
void fun2(void)
{
 int stack_i=300;
 int stack_j=400;
 printf("&stack_i=%p\n",&stack_i);
 printf("&stack_j=%p\n",&stack_j);
}
```

주소를 출력하기 위해 &stack_i를 출력했다
포인터이기 때문에 %p를 써줬다

```
void fun3(void)
{
 int *dynamic_p;
 dynamic_p=malloc(sizeof(100));
 printf("dynamic_p=%p\n",dynamic_p);
}
```

Dynamic_p 자체가 포인터이기 때문에 &를 써주지 않고 그대로 출력하였다

4.2

```
s2211051@oak:lab07$ gcc memory.c -o memory
s2211051@oak:lab07$ ./memory
&static_i=0x601048
&static_j=0x60104c
&stack_i=0x7ffe496fc6f0
&stack_j=0x7ffe496fc6f4
dynamic_p=0x10e3420
```

4.3

Stack은 높은 주소쪽에 메모리가 저장되고, 데이터는 낮은쪽, 그리고 heap은 중간에 저장된다. 그래서 stack은 메모리가 길다.