lect5 HW

시스템프로그래밍 1분반 12180626 성시열

1) [A char constant is an ascii number. A string constant is an address where it is stored in the string area.] Explain the result for following code.

```
-bash-4.2$ cd
-bash-4.2$ pwd
/home/sp5/12180626
-bash-4.2$ ls
d1  d2  d3  ex1  ex1.c  ex4.c  f1  f3  f4  f5  f6  f7  x  y.c
-bash-4.2$ vi ex1.c
```

```
#include <stdio.h>
#include <string.h>
void main()
        char x,y;
        x = 'a'; y = 97;
        printf("%d %c %d %c\n", x, x, y, y);
        char *x1 = "hello";
        printf("%s %p %s %p\n", x1, x1, "hello", "hello");
}
~
~
```

```
-bash-4.2$ gcc -o ex1 ex1.c
-bash-4.2$ ./ex1
97 a 97 a
hello 0x40064d hello 0x40064d
```

x와 y는 모두 문자형 변수로 선언되어있다. x에는 문자 a를 저장하고, y에는 97을 저장한다. y는 문자형 변수이므로 ASCII 코드 중 97에 해당하는 소문자 a가 y에 저장된다.
이후 printf를 통해 x 값을 정수로, x 값을 문자로, y 값을 정수로, y 값을 문자로 출력하면 "97 a 97 a"가 출력된다.
char * x1 = "hello"에서 "hello"는 문자열이 저장된 주소로 변환되고 그 주소값을 x1에 저장한다. *가 해당 변수에 주소값을 저장할 수 있게 한다.
이후 printf를 통해 x1 값을 문자열로, x1 값을 포인터 주소로, "hello" 값을 문자열로, "hello" 값을 포인터 주소로 출력하면 'hello [hello가 저장된 포인터주소] hello [hello가 저장된 포인터주소]"가 출력된다.

2) [A char constant is an ascii number] Try following code and explain the result.

```
#include <stdio.h>
void main()
        char x[10];
        x[0] = 'a'; x[1] = 'b'; x[2] = 'c'; x[3] = 'd'; x[4] = 'e';
        x[5] = 'f'; x[6] = 'g'; x[7] = 'h'; x[8] = 'i'; x[9] = 'j';
        int i;
        for(i=0; i<10; i++){
                printf("%d %c\n", x[i], x[i]);
        }
}
~
~
```

```
-bash-4.2$ vi ex5.c
-bash-4.2$ gcc -o ex5 ex5.c
-bash-4.2$ ./ex5
97 a
98 b
99 c
100 d
101 e
102 f
103 g
104 h
105 i
106 j
```

10칸의 문자를 넣을 수 있는 변수 x를 선언한 뒤
앞에서부터 차례로 a ~ j까지 입력하였다.
이후 반복문을 통해 0 번째 칸에 저장된 값의 정수형 결과와 문자형 결과를 출력한다.
그 결과
97 a
98 b
...
105 i
106 j
가 출력됨을 확인하였다. 이는 각 문자와 그에 맞는 ASCII 코드가 출력됨을 알 수 있었다.

3) Try below. Compare the result with that of Problem 2).
ex5.c를 수정하여 코드를 작성해보았다.

```
#include <stdio.h>
void main(){
        char x[10];
        int i;
        for(i=0; i<10; i++){
                x[i]=i+97;
        }
        for(i=0; i<10; i++){
                printf("%d %c\n", x[i], x[i]);
        }
}
~
~
```

```
-bash-4.2$ vi ex5.c
-bash-4.2$ gcc -o ex5 ex5.c
-bash-4.2$ ./ex5
97 a
98 b
99 c
100 d
101 e
102 f
103 g
104 h
105 i
106 j
```

2)에서는 배열의 각각 칸에 문자를 넣어줬다면 3)에서는 ASCII코드를 활용한 반복문을 통해

x[0]에 ASCII 코드 97에 해당하는 'a'를 x[1]에 ASCII 코드 98에 해당하는 'b'를 ,,, 반복하여 a ~ j까지 배열에 저장하였고, 출력하면 2)와 같은 결과가 나옴을 알 수 있다.

4) Declare a character array with 128 rooms. Store 0 to 127 in this array and print the corresponding character for each ascii code in the array. Find ASCII table in the Internet and confirm the results.

```
#include <stdio.h>
void main(){
        char x[128];
        int i;
        for(i=0; i<128; i++){
                x[i]=i;
        }
        for(i=0; i<128; i++){
                printf("%d %c %d\n", x[i], x[i], x[i]);
        }
}
~
~
```

```
0  0          30   30   60 < 60  90 Z 90
1  1          31   31   61 = 61  91 [ 91
2  2          32    32  62 > 62  92 \ 92
3  3          33 ! 33   63 ? 63  93 ] 93
4  4          34 " 34   64 @ 64  94 ^ 94
5  5          35 # 35   65 A 65  95 _ 95
6  6          36 $ 36   66 B 66  96 ` 96
7  7          37 % 37   67 C 67  97 a 97
8 8           38 & 38   68 D 68  98 b 98
9         9    39 ' 39   69 E 69  99 c 99
10            40 ( 40   70 F 70  100 d 100
 10           41 ) 41   71 G 71  101 e 101
11            42 * 42   72 H 72  102 f 102
    11        43 + 43   73 I 73  103 g 103
12            44 , 44   74 J 74  104 h 104
 12           45 - 45   75 K 75  105 i 105
 13           46 . 46   76 L 76  106 j 106
14  14        47 / 47   77 M 77  107 k 107
15  15        48 0 48   78 N 78  108 l 108
16  16        49 1 49   79 O 79  109 m 109 120 x 120
17  17        50 2 50   80 P 80  110 n 110 121 y 121
18  18        51 3 51   81 Q 81  111 o 111 122 z 122
19  19        52 4 52   82 R 82  112 p 112 123 { 123
20  20        53 5 53   83 S 83  113 q 113 124 | 124
21  21        54 6 54   84 T 84  114 r 114 125 } 125
22  22        55 7 55   85 U 85  115 s 115 126 ~ 126
23  23        56 8 56   86 V 86  116 t 116 127 127
24  24        57 9 57   87 W 87  117 u 117
25  25        58 : 58   88 X 88  118 v 118
26  26        59 ; 59   89 Y 89  119 w 119
27 7
28  28
29  29
```

3)처럼 ASCII 코드 97부터 10개의 값이 아닌 0부터 127까지 모든 ASCII 코드의 값들을 출력하였다. 실제 ASCII 코드와 비교해본 결과 일치함을 확인하였다.

5) [strlen] Read a string and display its length.

```
#include <stdio.h>
#include <string.h>
void main()
        char x[20];
        printf("Enter a string\n");
        scanf("%s", x);
        printf("The length is %d\n", strlen(x));
}
~
~
```

```
-bash-4.2$ vi ex5.c
-bash-4.2$ gcc -o ex5 ex5.c
-bash-4.2$ ./ex5
Enter a string
hello
The length is 5
```

문자형 배열을 저장하는 변수 x를 선언하고, 문자열을 입력받는다. strlen을 통해 입력받은 문자열을 길이를 출력한다.
"hello"라는 문자열을 입력하였을 때 문자열은 항상 뒤에 '0'이 붙으므로 6byte를 차지하지만 strlen에서는 사용자에게 보이는 길이인 5를 출력한다.

6) [A string is a char array ending with 0] Read a string and display each character in different lines.

```
#include <stdio.h>
#include <string.h>
void main()
        char x[20];
        printf("Enter a string\n");
        scanf("%s", x);
        int i;
        for(i=0; i < strlen(x); i++){
                printf("%c\n",x[i]);
        }
}
~
~
-bash-4.2$ vi ex5.c
-bash-4.2$ gcc -o ex5 ex5.c
-bash-4.2$ ./ex5
Enter a string
hello
h
e
l
l
o
```

문자열 각각의 문자가 한 줄에 한 글자씩 출력되게 하기 위해 반복문을 사용하였다. I는 0부터 문자열 길이만큼하여 입력된 문자열이 문자가 한 글자씩 각 줄마다 출력됨을 알 수 있다.

6-1) [A string is a char array ending with 0] Try below and explain the result. Use g++ to compile.

g++을 이용해 컴파일 하기 위해 코드를 C → C++로 변경해준다.
void main() → int main()
또한, C++에서는 반복문 for 안에서 변수를 선언해줄 수 있다.

```
#include <stdio.h>
#include <string.h>
int main()
        char x[10];
        strcpy(x, "hello");
        strcpy(x, "hi");
        for(int i=0; i < 10; i++){
                printf("%d ",x[i]);
        }
}
~
~
```

```
-bash-4.2$ vi ex5.c
-bash-4.2$ g++ -o ex5 ex5.c
-bash-4.2$ ./ex5
104 105 0 108 111 0 0 0 0 0 -bash-4.2$
```

배열 x를 선언하고, strcpy(x, "hello");를 통해 x[0] = h, x[1] = e, x[2] = l, x[3] = l, x[4] = o, x[5] = 0(NULL), 이 저장된다.
이후 strcpy(x, "hi");를 통해 x[0] = h, x[1] = i, x[2] = 0(NULL)이 저장된다.
최종 저장된 값: "hi0lo00000"
이렇게 저장된 x를 정수형태로 한칸씩 출력하면 다음과 같다.
h는 ASCII코드 104이고, i는 ASCII코드 105이고, 그 다음 0, 그 다음은 l인데, l은 ASCII코드 108이고, o는 ASCII코드 111이다. 이후에는 모두 0값이다.

7) [strlen, strcmp] Write a program that keeps reading a string, displaying its length, and checking whether it is "hello". If the input string is "hello", the program stops.

```
#include <stdio.h>
#include <string.h>
int main(){
        char x[20];
        for(;;){
                printf("Enter a string\n");
                scanf("%s", x);
                printf("You entered %s. length=%d\n", x, strlen(x));
                if(strcmp(x, "hello")==0){
                        printf("Yes it is hello. Bye.\n");
                        break;
                }
                else {
                        printf("No it is not hello\n");
                }
        }
}
~
~
```

```
-bash-4.2$ vi ex5.c
-bash-4.2$ g++ -o ex5 ex5.c
-bash-4.2$ ./ex5
Enter a string
hi
You entered hi. length=2
No it is not hello
Enter a string
hello
You entered hello. length=5
Yes it is hello. Bye.
```

hello가 입력되지 않을 경우에 무한 루프를 통해 계속 문자열을 받는 코드를 작성하였다. strcmp는 두 문자열을 비교하는 함수로 같으면 return 값이 0이 된다. 그러므로 strcmp(x, "hello")의 값이 0과 같다면 hello가 맞다고 하며 반복문을 빠져나온다. if(y=="hello")는 오류가 발생한다.

8) [strcpy] Read a string and copy it to three other string variables and change the first letter of them to 'a', 'b', and 'c', respectively, and display them.

```
#include <stdio.h>
#include <string.h>
int main()
        char x[20], a[20], b[20], c[20];
        printf("Enter a string\n");
        scanf("%s", x);
        strcpy(a, x); a[0] = 'a';
        strcpy(b, x); b[0] = 'b';
        strcpy(c, x); c[0] = 'c';
        printf("After copying and changing the first letter\n");
        printf("%s %s %s", a, b, c);
}
~
~
```

```
-bash-4.2$ vi ex5.c
-bash-4.2$ g++ -o ex5 ex5.c
-bash-4.2$ ./ex5
Enter a string
hello
After copying and changing the first letter
aello bello cello-bash-4.2$
```

문자를 입력받고 입력받은 문자의 첫 글자를 a, b, c로 바꿔주기 위해 배열 a, b, c에 입력받은 문자열 x를 복사하고, 배열 a, b, c 0번째 인덱스의 값을 a, b, c로 바꿔준다. 그 이후 a, b, c를 출력하면 변화된 값들이 출력된다.

9) [string constant] A string constant such as "hello" is an address. Explain the result of following code.

```c
#include <stdio.h>
#include <string.h>
void main(){
        char *x, *y, *z;
        x = "hello"; y = "hi"; z = "bye";
        printf("%s %s %s\n", x, y, z);
        printf("%p %p %p\n", x, y, z);
}
```

```
-bash-4.2$ vi ex5.c
-bash-4.2$ gcc -o ex5 ex5.c
-bash-4.2$ ./ex5
hello hi bye
0x400640 0x400646 0x400649
```

포인터 변수로 저장된 x, y, z에 각각 "hello", "hi", "bye"를 저장하고
해당하는 문자열 값과 포인터 주소값을 출력한다.
x의 포인터 주소 "400640"부터 6byte("hello"의 5byte + 문자열 마지막 NULL 1byte)를 차지하므로 y의 포인터 주소는 "400646"부터 시작된다.
y의 포인터 주소 "400646"부터 3byte("hi"의 2byte + 문자열 마지막 NULL 1byte)를 차지하므로 z의 포인터 주소는 "400649"부터 시작된다.
z의 포인터 주소 "400649"부터 4byte("bye"의 3byte + 문자열 마지막 NULL 1byte)를 차지한다.

10) [string constant is an address] Try below and explain why we have an error.

```c
#include <stdio.h>
#include <string.h>
void main(){
        char x[20];
        strcpy(x,"hello");
        x = "hello";
}
```

```
-bash-4.2$ vi ex5.c
-bash-4.2$ gcc -o ex5 ex5.c
ex5.c: In function 'main':
ex5.c:6:4: error: incompatible types when assigning to type 'char[20]' from type
 'char *'
```

위와 같은 코드를 작성한 뒤 컴파일을 하면 error가 발생한다.

x에 string을 담을 array를 선언하였다.
x="hello"를 통해 x값을 저장할 경우 문자열 "hello"는 "hello"가 저장된 주소 값을 뜻하고, 이러한 주소값은 array에 저장할 수 없으므로 error가 발생한다.

11) [You need memory space for strcpy] Try below and explain why we have an error. How can you fix it?

```c
#include <stdio.h>
#include <string.h>
void main()
        char *y;
        y = "hello1";
        strcpy(y,"hello2");
}
```

```
-bash-4.2$ vi ex5.c
-bash-4.2$ gcc -o ex5 ex5.c
-bash-4.2$ ./ex5
Segmentation fault (core dumped)
```

위와 같은 코드를 작성한 뒤 컴파일을 하면 error가 발생한다.

y는 주소값을 저장할 수 있도록 포인터 변수로 선언하였다.
y = "hello1"을 통해 y값을 저장할 경우, "hello1"은 "hello1"이 저장된 주소 값을 뜻하고, 이러한 주소값은 y에 저장할 수 있다.
하지만 string 함수 strcpy(y, "hello2")를 통해 값을 저장하면 문자열 그대로 y에 값을 저장하는데 이 때 y는 포인터 변수이므로 문자열 그대로 저장할 공간이 없다.
그 결과 코드를 실행시켜보면 error가 발생함을 확인하였다.
이를 수정하려면 strcpy(y, "hello2"); 코드를 지워준다.

```c
#include <stdio.h>
#include <string.h>
void main(){
        char *y;
        y = "hello1";
        printf("%s", y);
}
```

```
-bash-4.2$ gcc -o ex5 ex5.c
-bash-4.2$ ./ex5
hello1-bash-4.2$ vi ex5.c
```

컴파일 후 실행시켜본 결과 hello1이 출력됨을 확인하였다.

12) [You need memory space for scanf] Try below and explain why you have an error. Fix it.

```c
#include <stdio.h>
#include <string.h>
void main(){
        char *y;
        printf("enter a string\n");
        scanf("%s", y);
        printf("you entered %s\n", y);
}
```

```
-bash-4.2$ gcc -o ex5 ex5.c
-bash-4.2$ ./ex5
enter a string
hello
you entered (null)
```

y는 주소값을 저장할 수 있도록 포인터 변수로 선언하였다.
scanf를 통해 y의 값을 입력한다면 값이 주소값으로 저장되는 것이 아닌 하나의 문자열로 저장이 된다.
하지만 y는 포인터 변수이므로 문자열을 저장할 공간이 없으므로 scanf를 통해 입력된 문자열이 y에 저장되지 않는다.
그 결과 y의 값을 출력하였을 때 null이 출력된다.
이러한 문제는 y를 포인터 변수가 아닌 배열로 선언하면 해결된다.

```
#include <stdio.h>
#include <string.h>
void main(){
        char y[20];
        printf("enter a string\n");
        scanf("%s", y);
        printf("you entered %s\n", y);
}
~
~
```

```
-bash-4.2$ vi ex5.c
-bash-4.2$ gcc -o ex5 ex5.c
-bash-4.2$ ./ex5
enter a string
hello
you entered hello
```

코드를 수정하고 컴파일한 뒤 실행시켜 본 결과 입력된 문자열이 잘 출력됨을 확인하였다.

13) [char pointer array] Define a character pointer array and store/display strings as below.

```
#include <stdio.h>
#include <string.h>
void main(){
        char *x[10];
        x[0]="hi"; x[1]="bye"; x[2]="hello";
        printf("%s %s %s\n", x[0], x[1], x[2]);
}
~
~
```

```
-bash-4.2$ vi ex5.c
-bash-4.2$ gcc -o ex5 ex5.c
-bash-4.2$ ./ex5
hi bye hello
```

x를 포인터형 배열로 선언하였다.
x[0] = "hi"는 "hi"가 저장된 포인터 주소값을 x 배열 0번째 인덱스에 저장한다.
x[1] = "bye"는 "bye"가 저장된 포인터 주소값을 x 배열 1번째 인덱스에 저장한다.
x[2] = "hello"는 "hello"가 저장된 포인터 주소값을 x 배열 2번째 인덱스에 저장한다.
이후 각 배열에 저장된 값을 문자열 형태로 출력하면 "hi bye hello" 값이 출력된다.

14) [char pointer array, strcmp, new] Write a program that keeps reading strings and store them in a character pointer array. It stops when the user enters "end" and displays all strings entered so far. Use "new" to allocate memory and use g++ to compile.

```
#include <stdio.h>
#include <string.h>
int main(){
        char *x[100];
        int i;
        for(i = 0; ; i++){
                printf("Enter a string\n");
                x[i] = new char[20];
                scanf("%s", x[i]);
                if(strcmp(x[i], "end") == 0){
                        printf("String entered so far are\n");
                        break;
                }
        }
        for(int j = 0; j < i; j++){
                printf("%s ", x[j]);
        }
        printf("\n");
}
~
~
```

```
-bash-4.2$ vi ex5.c
-bash-4.2$ g++ -o ex5 ex5.c
-bash-4.2$ ./ex5
Enter a string
hi
Enter a string
aaa
Enter a string
bbb
Enter a string
end
String entered so far are
hi aaa bbb
```

x를 포인터형 배열로 선언하였다.
x의 0번째 인덱스부터 순서대로 문자열을 입력받는다. 이 때 x 배열의 각 원소를 동적 할당하여 각 원소가 가리키는 주소에 메모리를 할당해준다.
strcmp 함수에서 scanf를 통해 입력받은 값이 "end"라면 0이 return 되므로 이를 활용하여 return 값이 0일 때 아래 문장을 실행시키고 반복문을 탈출한다.
즉, 반복적으로 다양한 문자열을 입력하다가 끝내고 싶을 때 "end" 문자열을 scanf를 통해 입력한다. 그렇게 되면 여태 입력했던 문자열들을 쭉 출력한다.
C++ 프로그래밍을 사용하였으므로 g++을 통해 컴파일하고 실행한다. (new, int main() 등을 통해 C++임을 확인하였다.)
출력하면 end를 입력하기 전 문자열을 출력하는 것을 알 수 있다.

15) [gets, fgets] Read the same sentence with gets() and fgets() and explain the difference. (Ignore warning for gets. It is a security warning because gets can cause security problem.)

```
#include <stdio.h>
#include <string.h>
int main(){
        char x[100];
        //gets
        printf("Enter a sentence\n");
        gets(x);
        int slen = strlen(x);
        printf("sentence length after gets : %d\n", slen);
        for(int i = 0; i < slen; i++){
                printf("%x ", x[i]);
        }
        //fgets
        printf("\nenter the same sentence\n");
        fgets(x, 99, stdin); //max 99 char's
        slen = strlen(x);
        printf("sentence length after fgets : %d\n", slen);
        for(i = 0; i < slen; i++){
                printf("%x ", x[i]);
        }
}
~
~
-- INSERT --                                              21,2          All
```

```
-bash-4.2$ vi ex5.c
-bash-4.2$ g++ -o ex5 ex5.c
ex5.c: In function 'int main()':
ex5.c:8:2: warning: 'char* gets(char*)' is deprecated (declared at /usr/include/
stdio.h:640) [-Wdeprecated-declarations]
ex5.c:8:8: warning: 'char* gets(char*)' is deprecated (declared at /usr/include/
stdio.h:640) [-Wdeprecated-declarations]
-bash-4.2$ ./ex5
Enter a sentence
hello
sentence length after gets : 5
68 65 6c 6c 6f
enter the same sentence
hello
sentence length after fgets : 6
68 65 6c 6c 6f a
```

gets()와 fgets()의 차이를 알아보기 위해 코드를 작성하였다. gets를 통해 x에 문자열을 입력 받고, 문자열의 길이를 slen에 저장한다. "hello"를 입력하여 결과를 확인해보니 문자열 길이가 5임을 알 수 있다. x 배열의 각 인덱스를 %x를 통해 16진수로 표현된다.
'h'-68, 'e'-65, 'l'-6c, 'l'-6c, 'o'-6f 임을 알 수 있다.
fgets를 통해 x에 문자열을 입력 받고, 문자열의 길이를 slen에 저장한다. 위와 똑같이 "hello"를 입력하여 결과를 확인해보니 fgets는 마지막 공백을 포함하여 받으므로 strlen()의 결과가 6이 되고, 16진수로 표현할 때 마지막 공백이 a로 표현됨을 확인할 수 있다.

16) [strtok] Use strtok to extract words from a sentence and store them in an array. Display the number of words as below. Note that you need to copy the sentence to another string variable before doing strtok because strtok will destroy the original sentence.

```
#include <stdio.h>
#include <string.h>
int main(){
        char buf[256];
        char OG[256];
        char *token[256];
        int i = 0;
        printf("Enter a sentence\n");
        fgets(buf, 255, stdin);
        buf[strlen(buf)-1] = 0;
        strcpy(OG, buf);
        printf("You entered %s\n", buf);
        token[i] = strtok(buf, " ");
        while(1){
                printf("%s", token[i]);
                i++;
                token[i] = strtok(NULL, " ");
                if(token[i] == NULL) break;
        }
        printf("There were %d words\n", i);
        printf("The original sentence was:%s\n", OG);
}
~
-- INSERT --                                                    22,2          All
```

```
-bash-4.2$ ./ex5
Enter a sentence
aa bcd e e ff aa bcd hijk lmn al bcd
You entered aa bcd e e ff aa bcd hijk lmn al bcd
aabcdeeffaabcdhijklmnalbcdThere were 11 words
The original sentence was:aa bcd e e ff aa bcd hijk lmn al bcd
```

문자열을 받을 변수 buf 을 선언하고 fgets 를 활용하여 공백까지 입력을 받는다. strlen()을 통해 공백을 포함하여 문자열의 길이를 인식하므로 1 을 뺀 인덱스의 값이 마지막 인덱스인데, string 의 마지막 인덱스는 0 이므로 이 인덱스의 값을 0 으로 지정한다.
이후 buf 를 복사하여 배열 OG 에 붙여넣는다.
printf()를 통해 입력한 문자열을 확인한다.
이후 token 별로 값을 저장하기 위해 strtok 를 사용한다.
strtok 는 처음 실행할 때만 실제 변수 이름으로 하고 이후에는 NULL 로 하여 반복문을 작성한다. 값을 출력하면 token 별로 값을 출력하여 잘 저장이 되었는지 확인하였다. 이후 token 별로 구분하기 전 복사해두었던 OG 를 출력해보면 기존 문자열을 확인할 수 있다.

17) [char pointer array, new, strcmp] Write a program that keeps reading a name and stores it in a character pointer array until the user enters bye. The program should display all names after it sees "bye".

```
#include <stdio.h>
#include <string.h>
int main(){
        char *name[50];
        int i=0;
        while(1){
                printf("Enter a name\n");
                name[i] = new char[30];
                gets(name[i]);
                if(strcmp(name[i], "bye")==0) break;
                i++;
        }
        printf("There were %d names.\nThe names were\n", i);
        for(int j=0; j < i; j++){
                printf("%s\n", name[j]);
        }
}
~
~
```

```
-bash-4.2$ vi ex5.c
-bash-4.2$ g++ -o ex5 ex5.c
ex5.c: In function 'int main()':
ex5.c:9:3: warning: 'char* gets(char*)' is deprecated (declared at /usr/include/
stdio.h:640) [-Wdeprecated-declarations]
ex5.c:9:15: warning: 'char* gets(char*)' is deprecated (declared at /usr/include
/stdio.h:640) [-Wdeprecated-declarations]
-bash-4.2$ ./ex5
Enter a name
kim han kook
Enter a name
park dong il
Enter a name
hong gil dong
Enter a name
bye
There were 3 names.
The names were
kim han kook
park dong il
hong gil dong
```

문자열을 저장할 name 이라는 포인터 배열을 선언한다. 반복문을 통해 이름을 반복적으로 입력받고, "bye"라는 문자열일 때 반복문을 빠져나온다. 반복적으로 입력받은 값은 name 이라는 포인터 배열에 저장된다.
printf()를 통해 입력받은 값의 개수를 출력하고
for()문을 통해 포인터 배열에 저장해둔 값을 출력한다.

18) [There is a hidden 0 at the end of a string] Try below and explain why it behaves strange. How can you fix it?

```c
#include <stdio.h>
#include <string.h>
int main(){
        int x1;
        char x2[12];
        x1 = 33;
        strcpy(x2, "abcdefghijkl");
        printf("%d %s\n", x1, x2);
}
```

```
-bash-4.2$ vi ex5.c
-bash-4.2$ g++ -o ex5 ex5.c
-bash-4.2$ ./ex5
0 abcdefghijkl
```

char x2[12];를 통해 12칸의 array를 생성하였고, strcpy()를 통해 12개의 문자를 붙여넣는다. string은 마지막에 0이 들어가야하기 때문에 생성한 array보다 더 많은 값이 들어간다. 이를 stack overflow라고 하고 이로 인해 x1의 값이 0이 출력되는 오류가 발생하였다.

```c
#include <stdio.h>
#include <string.h>
int main(){
        int x1;
        char x2[13];
        x1 = 33;
        strcpy(x2, "abcdefghijkl");
        printf("%d %s\n", x1, x2);
}
```

```
-bash-4.2$ vi ex5.c
-bash-4.2$ g++ -o ex5 ex5.c
-bash-4.2$ ./ex5
33 abcdefghijkl
```

오류를 해결하기 위해 배열을 13칸으로 하여 실행하면 x1이 33이 출력되는 것을 알 수 있다.

19) [You need memory space for strcpy] What is wrong with the following program? How can you fix it?

```c
#include <stdio.h>
#include <string.h>
int main(){
        char *strarr[10]={NULL};
        strarr[0] = "hello";
        strcpy(strarr[1], "bye");
        printf("%s %s\n", strarr[0], strarr[1]);
}
```

```
-bash-4.2$ vi ex5.c
-bash-4.2$ g++ -o ex5 ex5.c
ex5.c: In function 'int main()':
ex5.c:5:14: warning: deprecated conversion from string constant to 'char*' [-Wwr
ite-strings]
-bash-4.2$ ./ex5
Segmentation fault (core dumped)
```

strarr라는 포인터 배열 변수를 선언하였다. 이 변수에는 주소 값이 저장되는데, strcpy를 사용할 때 strarr[1]에 문자열 값 자체를 넣으므로 에러가 발생한다.

```
#include <stdio.h>
#include <string.h>
int main(){
        char *strarr[10]={NULL};
        strarr[0] = "hello";
        strarr[1] = "bye";
        printf("%s %s\n", strarr[0], strarr[1]);
}
~
~
```

```
-bash-4.2$ vi ex5.c
-bash-4.2$ g++ -o ex5 ex5.c
ex5.c: In function 'int main()':
ex5.c:5:14: warning: deprecated conversion from string constant to 'char*' [-Wwr
ite-strings]
ex5.c:6:14: warning: deprecated conversion from string constant to 'char*' [-Wwr
ite-strings]
-bash-4.2$ ./ex5
hello bye
```

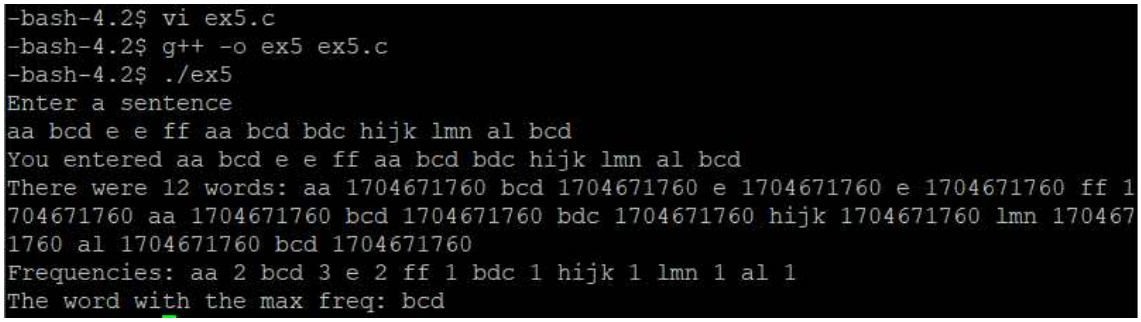
위 같은 에러를 수정하기 위해 strarr[1] = "bye"를 활용하여 "bye"가 저장된 주소값을 strarr[1]에 저장한다.

출력하면 "hello bye"가 출력되는 것을 확인하였다.

20) [char pointer array, new, strcmp] Write a program that reads a long sentence and displays the frequency of each word as below. It also prints the word that has the maximum frequency.

```c
#include <stdio.h>
#include <string.h>

int tokenize(char buf[], char *token[]);
void compute(char *token[], int fre[], int words);

int main(){
        printf("Enter a sentence\n");
        char buf[256];
        char *token[256] = {NULL, };
        int fre[256] = {0, };
        fgets(buf, 99, stdin);

        buf[strlen(buf) - 1] = '\0';
        printf("You entered %s\n", buf);

        int words_of_token = tokenize(buf, token);
        printf("There were %d words: ", words_of_token);
        for(int i=0;i<words_of_token;i++){
                printf("%s %d ", token[i]);
        }
        compute(token, fre, words_of_token);
}
```

```c
int tokenize(char buf[], char* token[]){
        int i = 0;
        token[i] = strtok(buf, " ");
        while(1){
                i++;
                token[i] = strtok(NULL, " ");
                if(token[i] == NULL) break;
        }
        return i; //number of tokens
}

void compute(char *token[], int fre[], int words){
        for(int i = 0; i < words; i++){
                for(int j = 0; j < words; j++){
                        if(strcmp(token[i], token[j]) == 0){
                                fre[j] = fre[j] + 1; //frequency
                                break;
                        }
                }
        }
        printf("\nFrequencies: ");
        for (int k = 0; k < words; k++){
                if(fre[k] != 0) printf("%s %d ", token[k], fre[k]);
        }

        //max frequency
        int max = 0, index_max = 0;
        for (int m = 0; m < words; m++){
                if(max < fre[m]){
                        max = fre[m];
                        index_max = m;
                }
        }
        printf("\nThe word with the max freq: %s\n", token[index_max]);
}
                                                                59,1          Bot
```



문자열을 공백단위로 나눠서 저장해주는 tokenize 함수, 빈도수를 계산하는 compute 함수 두 가지 함수를 생성한 뒤 코드를 구성한다.

함수 프로토타입을 작성하여 main 함수에서 tokenize 함수와 compute 함수를 사용할 수 있게 하였다.

tokenize 함수는 앞서 16)번 문제에서 작성했던 strtok() 함수를 활용해 tokenize 함수를 완성한다.

compute 함수는 이중 for 문을 활용하여 token 끼리 값을 비교하여 같은 횟수만큼 counting 한다. 이후 같은 횟수(빈도수)가 가장 높은 값을 max 에 저장하고 그 때의 index 를 max_index 로 저장한다. 이후에 max_index 에 해당하는 값을 출력하여 가장 빈도수가 높은 문자를 출력한다.