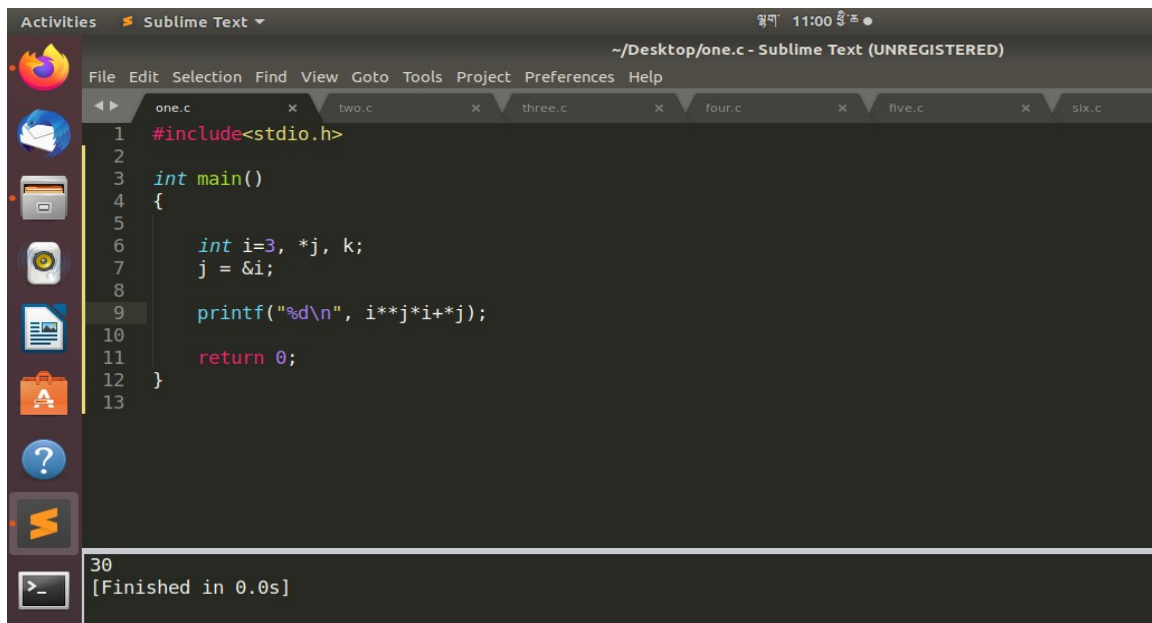


LAB 4

**Submitted by:
Samten Wangmo
12190073
Group “A”**

Question 1: What will be the output of the following program? Note down your understanding of every program, in few sentences.



```
1 #include<stdio.h>
2
3 int main()
4 {
5
6     int i=3, *j, k;
7     j = &i;
8
9     printf("%d\n", i**j*i+j);
10
11     return 0;
12 }
13
```

30
[Finished in 0.0s]

Justification: i^{**} will point the pointer towards the pointer and print the value at the pointed variable. The value printed will be 3 since it will be pointing towards the (i).

$j = \&i$, it says that the value of j will be stored as the address of (i).

j^{*} will points toward the value of (i), which is =3.

i will print the value of (i).

Thus, the output will be $3*3*3+3 = 30$.

The screenshot shows a Sublime Text editor window with the title bar indicating the file path is `~/Desktop/two.c`. The editor has multiple tabs open, with `two.c` selected. The code in `two.c` is as follows:

```
1 #include<stdio.h>
2 int main()
3 {
4     int x=30, *y, *z;
5     y=&x;
6     //Address of x depends from pc-pc.
7     //However, integer is 4 byte size
8
9     z=y;
10    *y++=*z++;
11    x++;
12    printf("x=%d, y=%d, z=%d\n", x, y, z);
13
14    return 0;
15 }
```

A warning message is displayed below the code: `warning: format '%d' expects argument of type 'int', but argument 3 has type 'int *' [-Wformat]`.

Below the editor, a terminal window shows the output of the program:

```
/home/user/Desktop/two.c:12:25: warning: format '%d' expects argument of type 'int', but argument 4 has type 'int *'
[-Wformat=]
printf("x=%d, y=%d, z=%d\n", x, y, z);
                        ^~
                        %ls
x=31, y=1232646488, z=1232646488
[Finished in 0.1s]
```

Justification: Value of $x = 30$.

Value of y will be stored as the address of x .

The value of y will be the value of x .

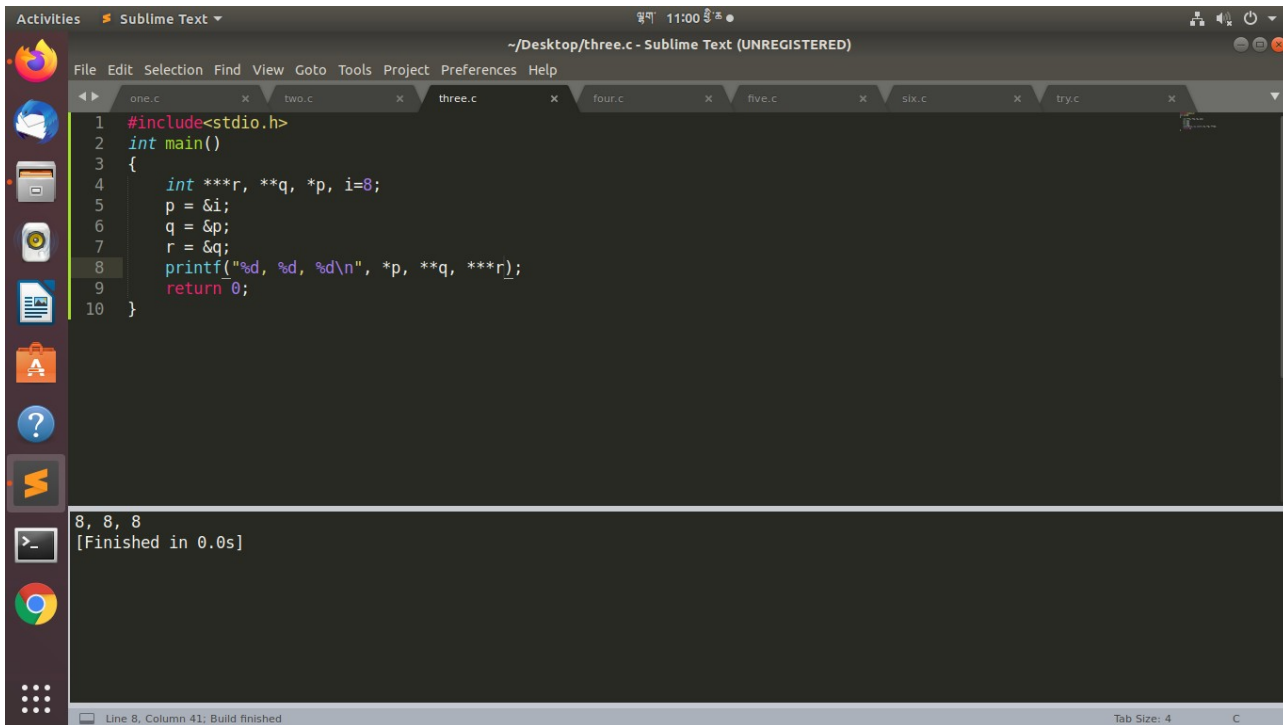
Since, $*y$ will point toward the value of x and $*y = *z$, so $*z$ will also points toward the value of x .

In the condition : $*y++ = *z ++$; $x++$, The value of x will be incremented by 1 which is $30+1 = 31$.

$y = \&x$;

$z = y$,

y will print the address of x and since $z = y$, it will also print the address of x .



The screenshot shows a Sublime Text editor window titled "Sublime Text (UNREGISTERED)" with the file path "~/Desktop/three.c". The editor has several tabs open: one.c, two.c, three.c, four.c, five.c, six.c, and try.c. The active tab is "three.c", which contains the following C code:

```
1 #include<stdio.h>
2 int main()
3 {
4     int ***r, **q, *p, i=8;
5     p = &i;
6     q = &p;
7     r = &q;
8     printf("%d, %d, %d\n", *p, **q, ***r);
9     return 0;
10 }
```

Below the code editor, the output of the program is displayed in a terminal window:

```
8, 8, 8
[Finished in 0.0s]
```

The status bar at the bottom indicates "Line 8, Column 41: Build finished" and "Tab Size: 4 C".

***r = pointer to a pointer to a pointer,

**q = pointer to a pointer.

*p = pointer.

While printing output:

*p will print the values at the address of I which is = 8.

**q will point toward the address of p and then towards the address of I which is = 8.

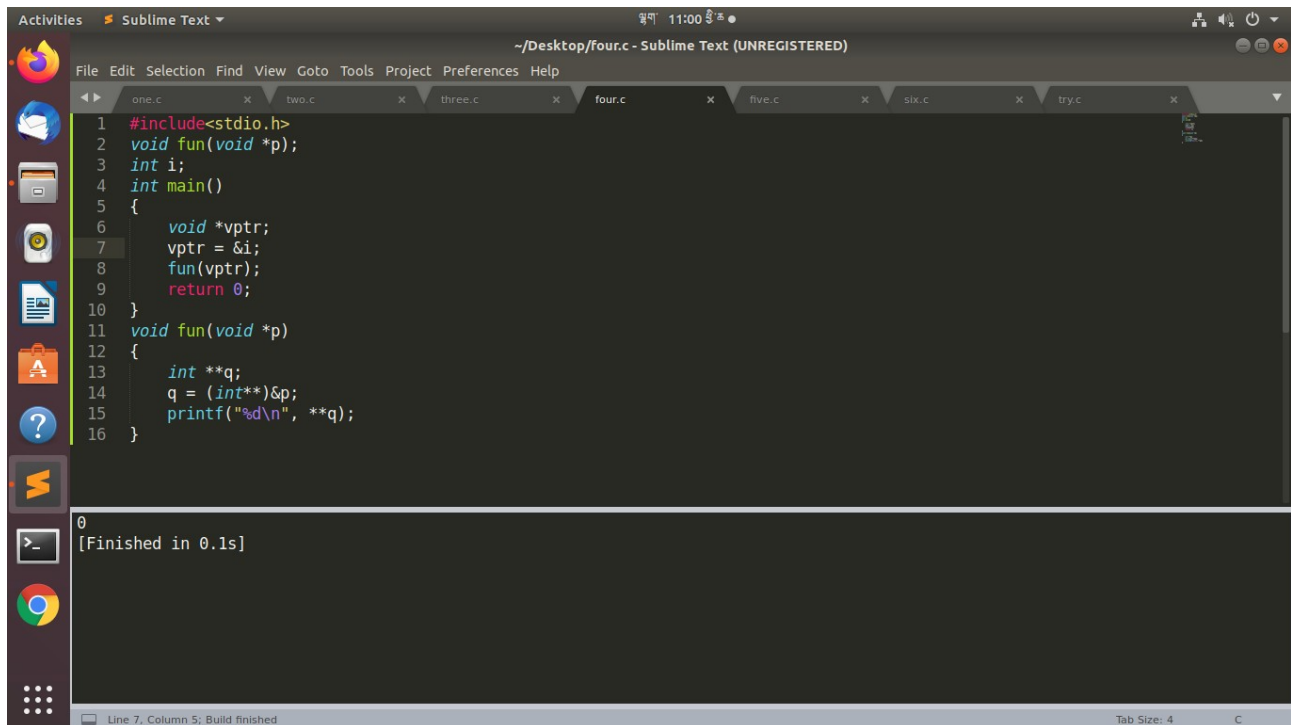
***r will also work same as first as it will point towards the address of q and then to the address of p and finally to the address of I and prints the value at the address of I which is = 8.

Justification:

Int i is global variable.

The value of global variable is 0 by default.

Since all the pointers are pointing towards the global variable, value 0 will be printed.



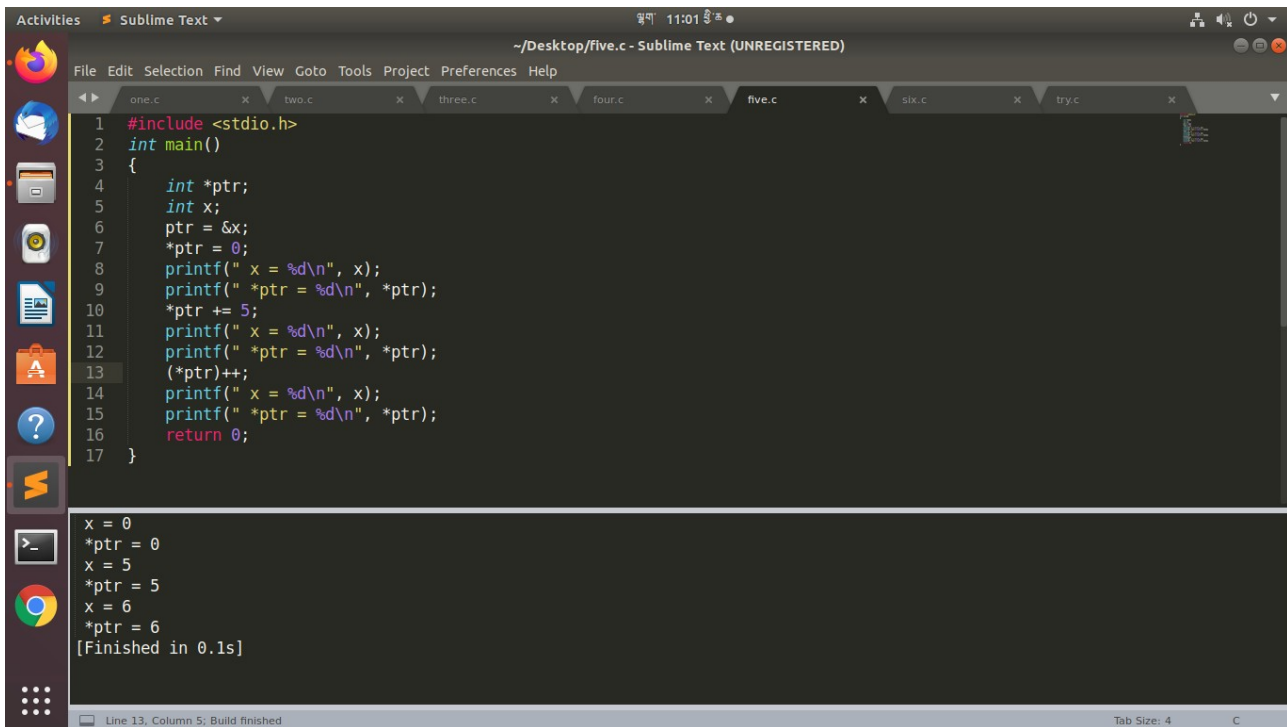
The screenshot shows a Sublime Text editor window titled "Sublime Text" with a menu bar (File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help) and a toolbar. The editor has several tabs open: one.c, two.c, three.c, four.c, five.c, six.c, and try.c. The active tab is "four.c", which contains the following C code:

```
1 #include<stdio.h>
2 void fun(void *p);
3 int i;
4 int main()
5 {
6     void *vptr;
7     vptr = &i;
8     fun(vptr);
9     return 0;
10 }
11 void fun(void *p)
12 {
13     int **q;
14     q = (int**) &p;
15     printf("%d\n", **q);
16 }
```

Below the editor, a terminal window shows the output of the program:

```
0
[Finished in 0.1s]
```

The status bar at the bottom indicates "Line 7, Column 5; Build finished" and "Tab Size: 4 C".



The screenshot shows a Sublime Text editor window titled "Sublime Text (UNREGISTERED)" with the file path "~/Desktop/five.c". The editor has several tabs open: "one.c", "two.c", "three.c", "four.c", "five.c", "six.c", and "try.c". The "five.c" tab is active, displaying the following C code:

```
1 #include <stdio.h>
2 int main()
3 {
4     int *ptr;
5     int x;
6     ptr = &x;
7     *ptr = 0;
8     printf(" x = %d\n", x);
9     printf(" *ptr = %d\n", *ptr);
10    *ptr += 5;
11    printf(" x = %d\n", x);
12    printf(" *ptr = %d\n", *ptr);
13    (*ptr)++;
14    printf(" x = %d\n", x);
15    printf(" *ptr = %d\n", *ptr);
16    return 0;
17 }
```

Below the code editor, the output of the program is displayed in a terminal-like window:

```
x = 0
*ptr = 0
x = 5
*ptr = 5
x = 6
*ptr = 6
[Finished in 0.1s]
```

The status bar at the bottom indicates "Line 13, Column 5; Build finished" and "Tab Size: 4 C".

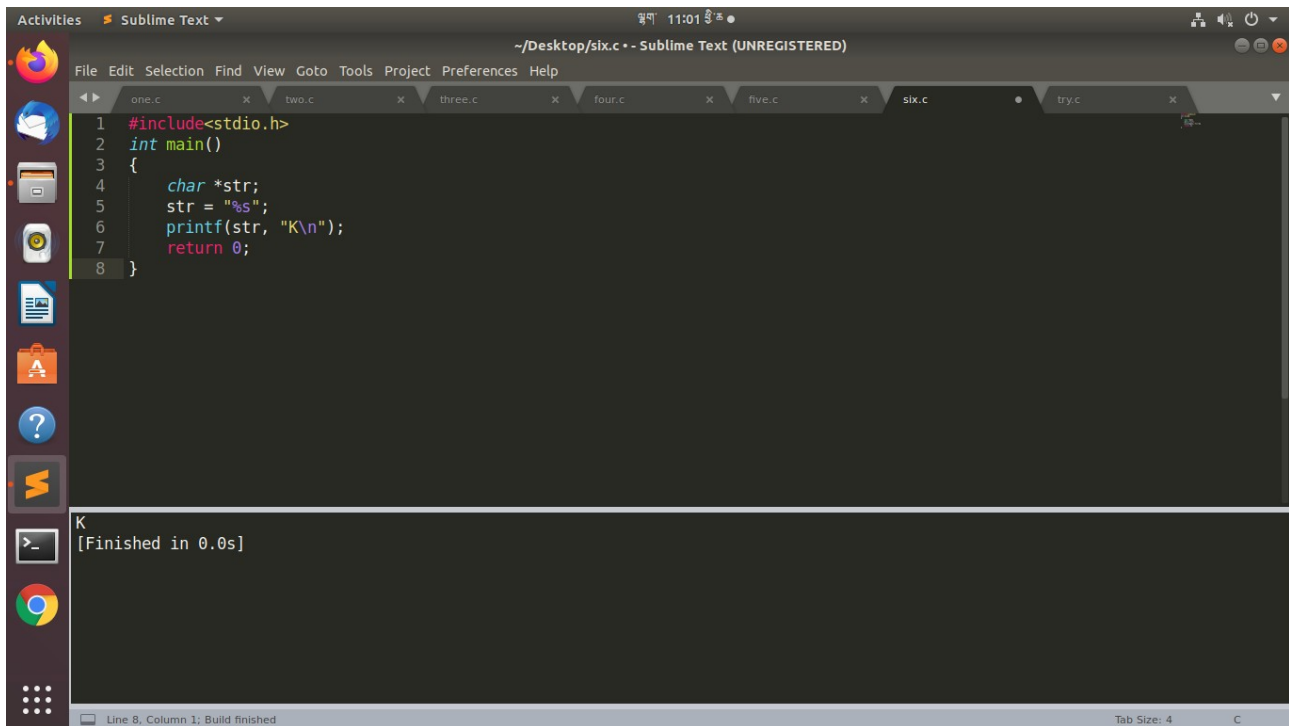
Justification:

`ptr = &x;`
ptr stores the value at the address of x

`*ptr = 0;`
Value at the address of x where the ptr is pointing will be zero which means that $x = 0$ since pointer is pointing towards the value of address x.

Since now, `*ptr += 5;`
It will print $x = 5$ and `*ptr = 5`.

Again it say that value at the pointer is incremented by one so it will print $5+1$ which is = 6.



```
1 #include<stdio.h>
2 int main()
3 {
4     char *str;
5     str = "%s";
6     printf(str, "K\n");
7     return 0;
8 }
```

K

[Finished in 0.0s]

Line 8, Column 1: Build finished

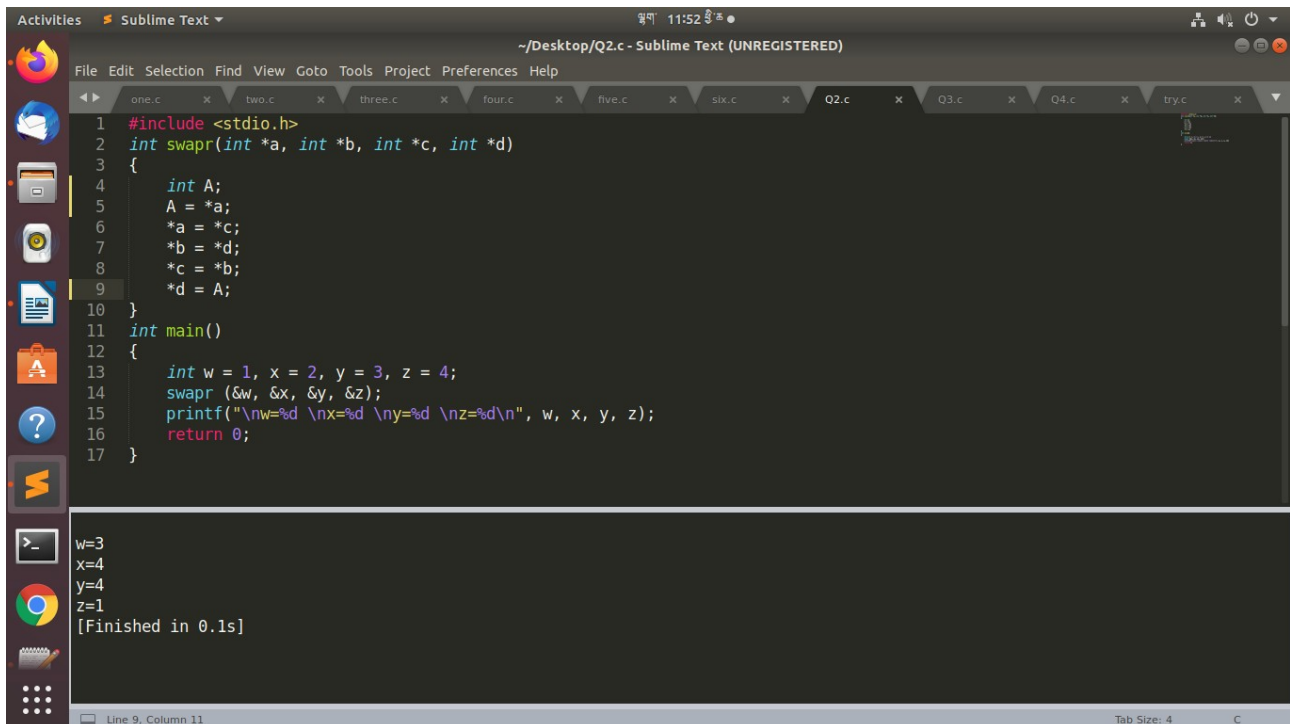
Justification:

str = “%s” which means that the corresponding argument will be treated as string.

In this question, the position of the value of str and the “%s” format specifier is changed to make us think for some minute.

The value printed will be only k.

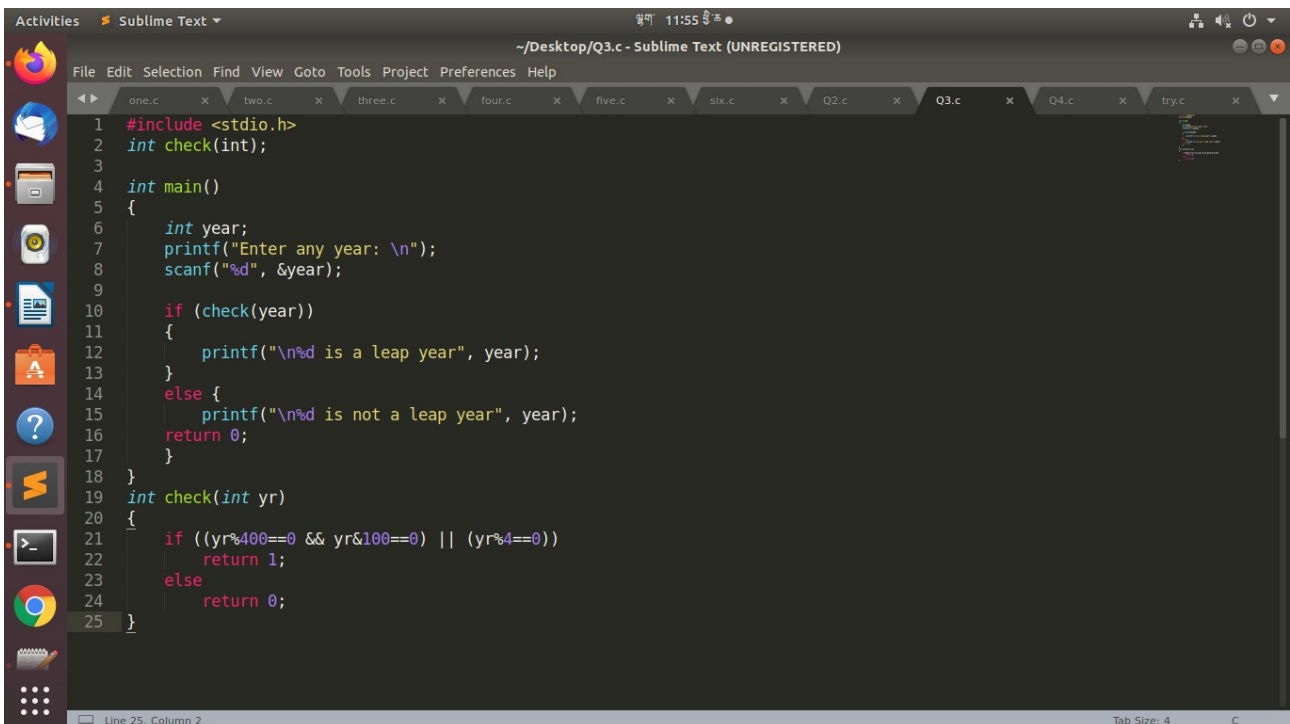
2) Write a C Program to swap 4-different elements using Call by Reference.



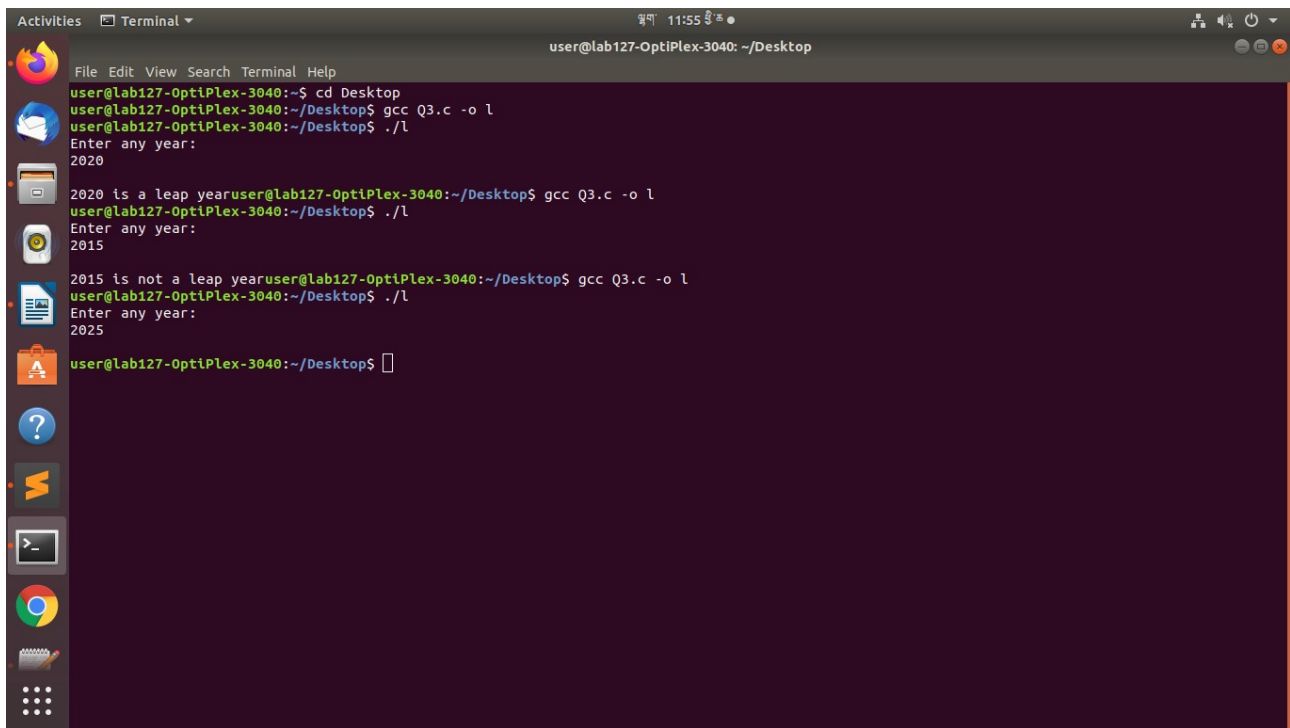
```
1 #include <stdio.h>
2 int swapr(int *a, int *b, int *c, int *d)
3 {
4     int A;
5     A = *a;
6     *a = *c;
7     *b = *d;
8     *c = *b;
9     *d = A;
10 }
11 int main()
12 {
13     int w = 1, x = 2, y = 3, z = 4;
14     swapr (&w, &x, &y, &z);
15     printf("\nw=%d \nx=%d \ny=%d \nz=%d\n", w, x, y, z);
16     return 0;
17 }
```

w=3
x=4
y=4
z=1
[Finished in 0.1s]

3) WAP a program to find if the Year entered by the user through keyboard is a leap year or not. Apply Call by Reference concept.



```
1 #include <stdio.h>
2 int check(int);
3
4 int main()
5 {
6     int year;
7     printf("Enter any year: \n");
8     scanf("%d", &year);
9
10    if (check(year))
11    {
12        printf("\n%d is a leap year", year);
13    }
14    else {
15        printf("\n%d is not a leap year", year);
16        return 0;
17    }
18 }
19 int check(int yr)
20 {
21     if ((yr%400==0 && yr%100==0) || (yr%4==0))
22         return 1;
23     else
24         return 0;
25 }
```

A terminal window titled "Terminal" showing the execution of a C program. The user is at the prompt "user@lab127-OptiPlex-3040: ~/Desktop". The program prompts "Enter any year:" and the user enters "2020". The output is "2020 is a leap year". The user then enters "2015", and the output is "2015 is not a leap year". Finally, the user enters "2025", and the output is "2025 is not a leap year". The terminal shows the compilation command "gcc Q3.c -o l" and the execution command "./l".

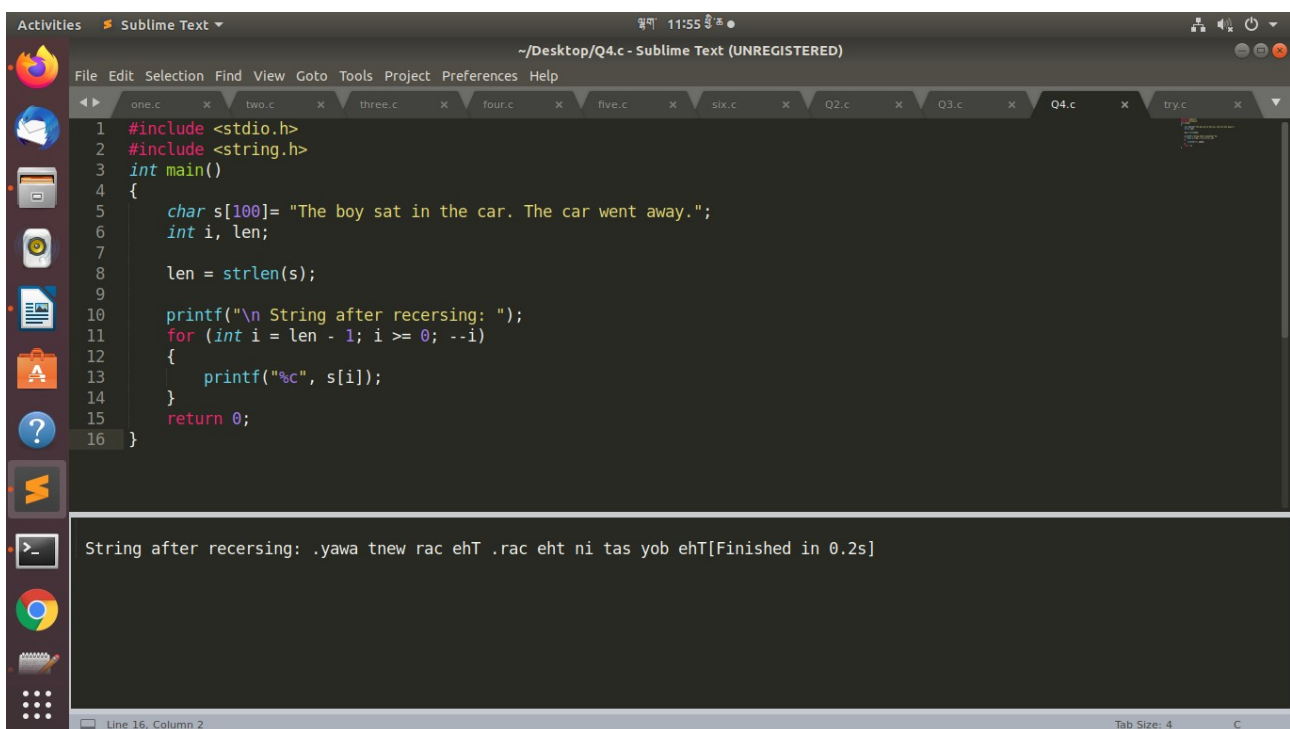
```
user@lab127-OptiPlex-3040:~$ cd Desktop
user@lab127-OptiPlex-3040:~/Desktop$ gcc Q3.c -o l
user@lab127-OptiPlex-3040:~/Desktop$ ./l
Enter any year:
2020
2020 is a leap year
user@lab127-OptiPlex-3040:~/Desktop$ gcc Q3.c -o l
user@lab127-OptiPlex-3040:~/Desktop$ ./l
Enter any year:
2015
2015 is not a leap year
user@lab127-OptiPlex-3040:~/Desktop$ gcc Q3.c -o l
user@lab127-OptiPlex-3040:~/Desktop$ ./l
Enter any year:
2025
2025 is not a leap year
user@lab127-OptiPlex-3040:~/Desktop$
```

4) WAP with inbuilt C-function that reverses the input text one line at a time.e.g.
input : The boy sat in the car. The car went away.

Output:

rac eht ni yob eht.

Yawa tnew rac ehT.



A Sublime Text editor window titled "Q4.c - Sublime Text (UNREGISTERED)" showing a C program. The code includes `<stdio.h>` and `<string.h>`, defines a string `s` with the value "The boy sat in the car. The car went away.", and uses `strlen` to get its length. It then prints the string in reverse order using a loop. The output in the terminal is "String after recersing: .yawa tnew rac ehT .rac eht ni tas yob ehT[Finished in 0.2s]".

```
#include <stdio.h>
#include <string.h>
int main()
{
    char s[100]= "The boy sat in the car. The car went away.";
    int i, len;

    len = strlen(s);

    printf("\n String after recersing: ");
    for (int i = len - 1; i >= 0; --i)
    {
        printf("%c", s[i]);
    }
    return 0;
}
```

String after recersing: .yawa tnew rac ehT .rac eht ni tas yob ehT[Finished in 0.2s]