Practical - 2

Aim : Understanding of macro facilities which will demonstrate the use of following preprocessor directive of C.

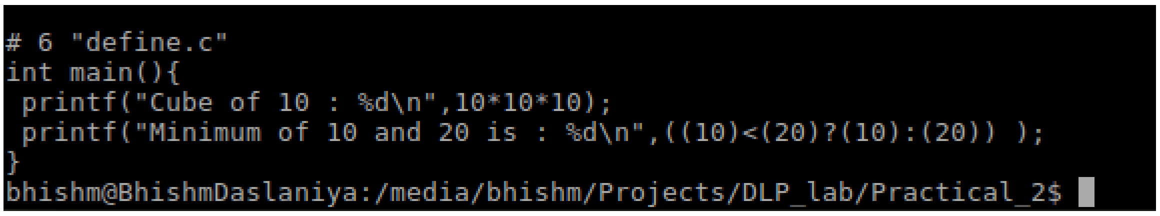
1. #define : A macro is a segment of code which is replaced by the value of macro. Macro is defined by #define directive.
There are two types of macros:

1.Object-like Macros: #define PI 3.1415

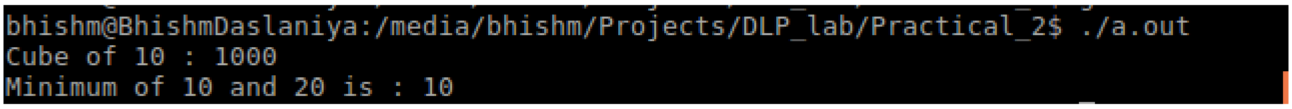
2.Function-like Macros: #define MIN(a,b) ((a)<(b)?(a):(b))

Code:
```
#include<stdio.h>
#define SQRT(x) x*x
#define CUBE(x) SQRT(x)*x
#define MIN(a,b) ((a)<(b)?(a):(b))

int main(){
        printf("Cube of 10 : %d\n",CUBE(10));
        printf("Minimum of 10 and 20 is : %d\n",MIN(10,20) );
}
```
Output:



```
# 6 "define.c"
int main(){
 printf("Cube of 10 : %d\n",10*10*10);
 printf("Minimum of 10 and 20 is : %d\n",((10)<(20)?(10):(20)) );
}
bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$
```



```
bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$ ./a.out
Cube of 10 : 1000
Minimum of 10 and 20 is : 10
```

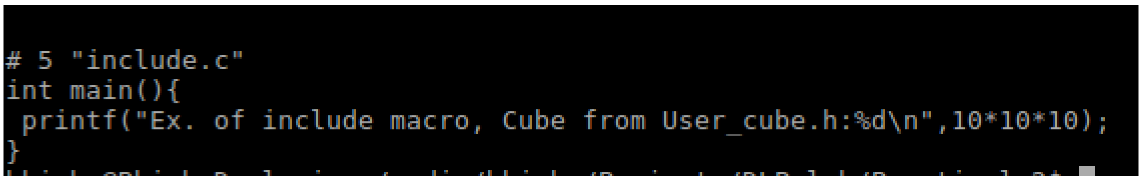
2. #include: The #include preprocessor directive is used to paste code of given file into current file.

Code:

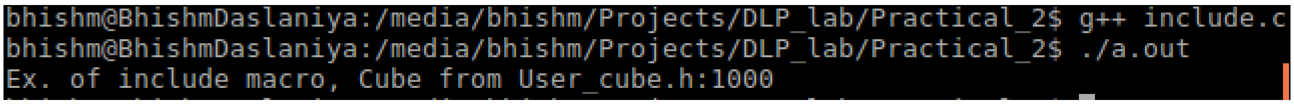
User_cube.h:
```
#define SQRT(x) x*x
#define CUBE(x) SQRT(x)*x
```

include.c:
```
#include<stdio.h>
#include<math.h>
#include "User_cube.h"
int main(){
        printf("Ex. of include macro, Cube from User_cube.h:%d\n",CUBE(10));
}
```

Output:



```
# 5 "include.c"
int main(){
 printf("Ex. of include macro, Cube from User_cube.h:%d\n",10*10*10);
}
```



```
bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$ g++ include.c
bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$ ./a.out
Ex. of include macro, Cube from User_cube.h:1000
```

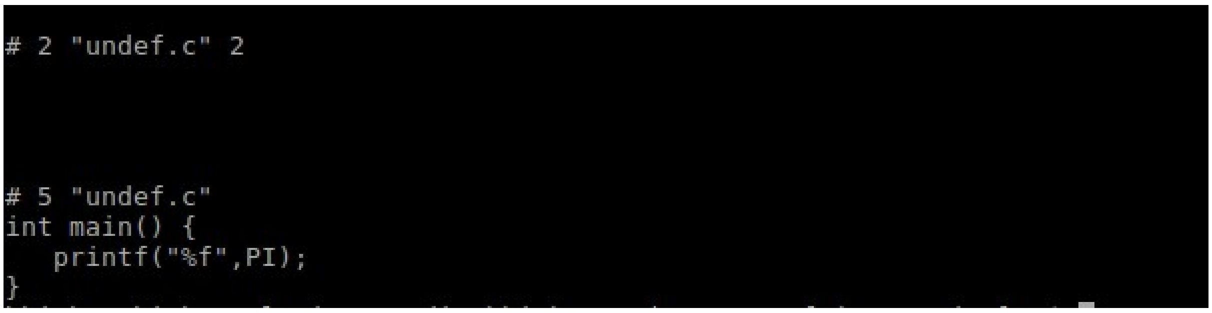
3. #undef: It is useful for undefine a macro means to cancel its defination.

Code:

```
#include <stdio.h>
#define PI 3.1415
#undef PI

int main() {
  printf("%f",PI);
}
```

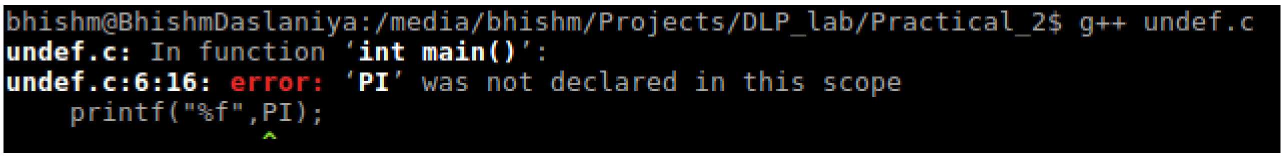
Output:



```
# 2 "undef.c" 2




# 5 "undef.c"
int main() {
   printf("%f",PI);
}
```



```
bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$ g++ undef.c
undef.c: In function 'int main()':
undef.c:6:16: error: 'PI' was not declared in this scope
    printf("%f",PI);
                ^
```

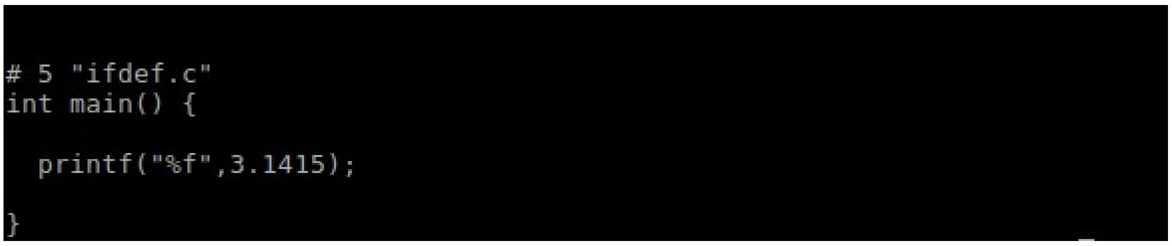
4. #ifdef: The #ifdef preprocessor directive checks if macro is defined by #define. If yes, it executes the code.

Code:

```
#include <stdio.h>
#define PI 3.1415
int main() {
        #ifdef PI
                printf("%f",PI);
```

```
	#endif
}
```

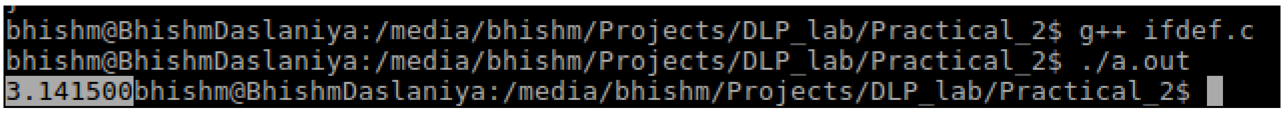
Output:



```
# 5 "ifdef.c"
int main() {

  printf("%f",3.1415);

}
```



```
bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$ g++ ifdef.c
bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$ ./a.out
3.141500bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$
```

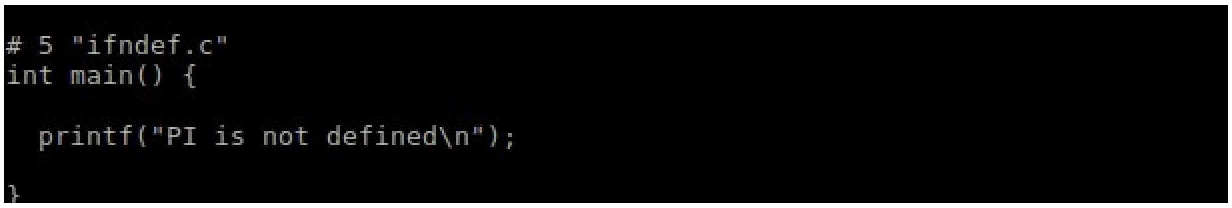
5. #ifndef: The #ifndef preprocessor directive checks if macro is not defined by #define. If yes, it executes the code.

Code:

```
#include <stdio.h>
#define PI 3.1415
#undef PI

int main() {
	#ifndef PI
		printf("PI is not defined\n");
	#endif
}
```

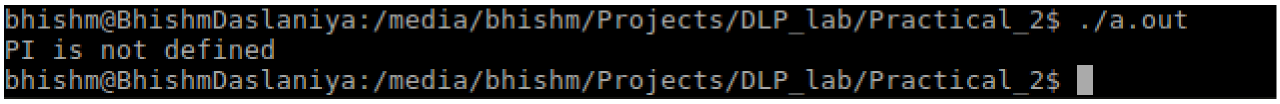
Output:



```
# 5 "ifndef.c"
int main() {

  printf("PI is not defined\n");

}
```



```
bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$ ./a.out
PI is not defined
bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$
```

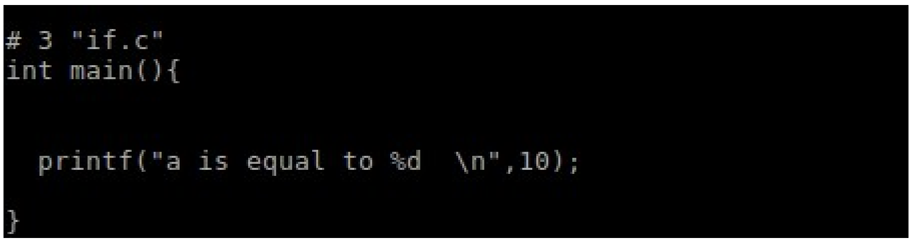
6. #if: The #if preprocessor directive evaluates the expression or condition. If condition is true, it executes the code.

Code:

```
#include<stdio.h>
#define a 10
int main(){
	#if a==10
```

```
                printf("a is equal to %d  \n",a);
        #endif
}
```

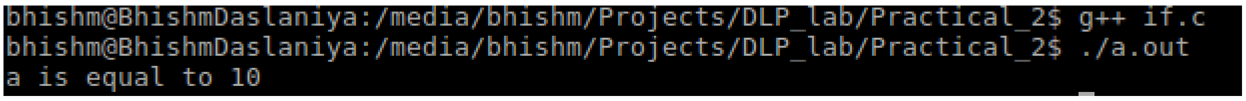
Output:



```
# 3 "if.c"
int main(){


  printf("a is equal to %d  \n",10);

}
```



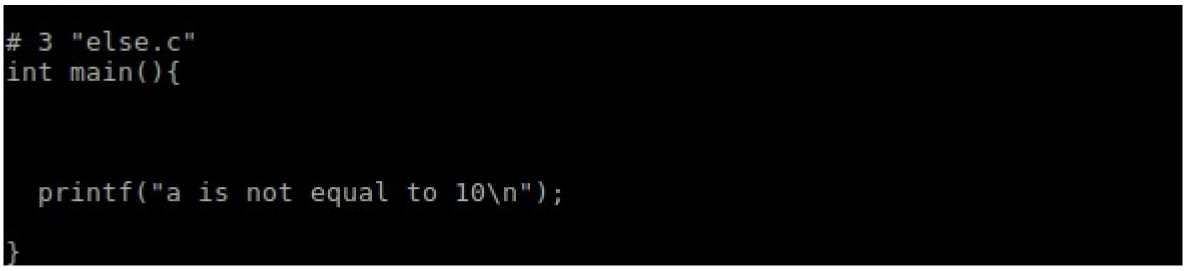
```
bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$ g++ if.c
bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$ ./a.out
a is equal to 10
```

7. #else: The #else preprocessor directive evaluates the expression or condition if condition of #if is false. It can be used with #if, #elif, #ifdef and #ifndef directives.

Code:

```
#include<stdio.h>
#define a 11
int main(){
        #if a==10
                printf("a is equal to %d  \n",a);
        #else
                printf("a is not equal to 10\n");
        #endif
}
```

Output:

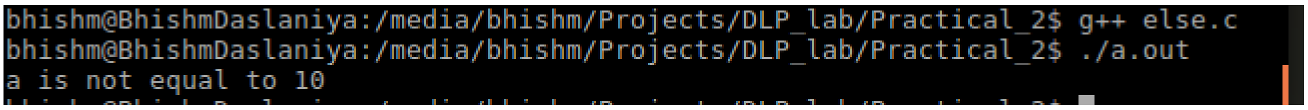


```
# 3 "else.c"
int main(){



  printf("a is not equal to 10\n");

}
```



```
bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$ g++ else.c
bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$ ./a.out
a is not equal to 10
```

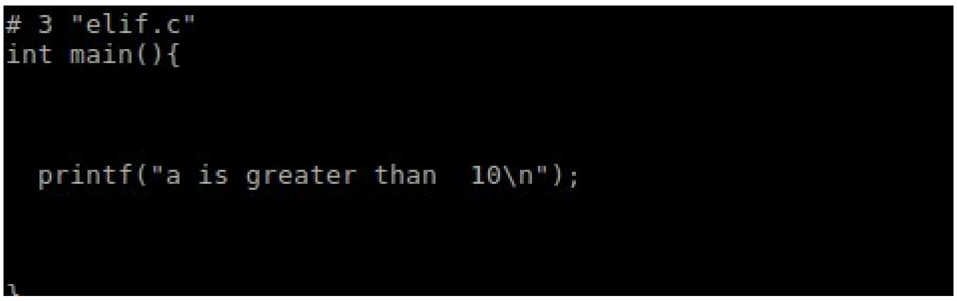
8. #elif: It is useful for if-else ladder kind of statements.

Code:

```
#include<stdio.h>
#define a 11
```

```
int main(){
        #if a==10
                printf("a is equal to %d  \n",a);
        #elif a > 10
                printf("a is greater than  10\n");
        #else
                printf("a is less than  10\n");
        #endif
}
```

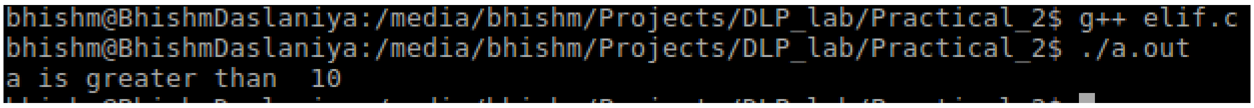
Output:



```
# 3 "elif.c"
int main(){



  printf("a is greater than  10\n");
```



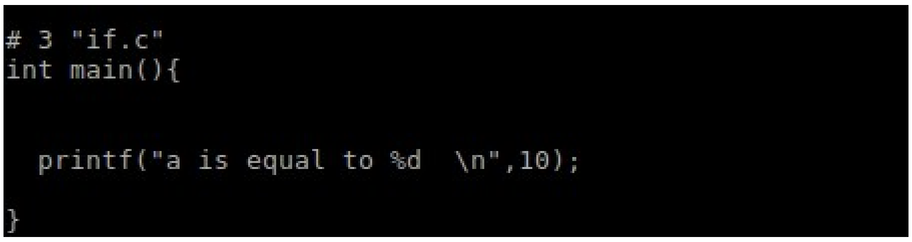
```
bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$ g++ elif.c
bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$ ./a.out
a is greater than  10
```

9. #endif: It is useful for complete the #if preprocessor directive means complete the if block.

Code:

```
#include<stdio.h>
#define a 10
int main(){
        #if a==10
                printf("a is equal to %d  \n",a);
        #endif
}
```

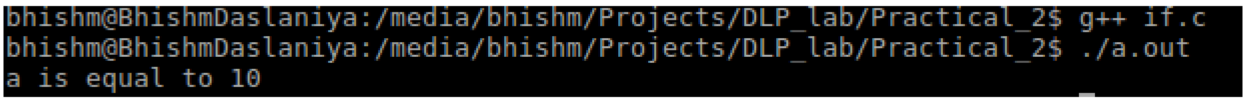
Output:



```
# 3 "if.c"
int main(){


  printf("a is equal to %d  \n",10);

}
```



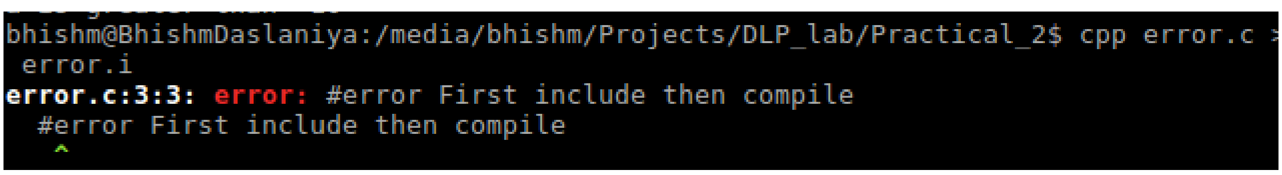
```
bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$ g++ if.c
bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$ ./a.out
a is equal to 10
```

10.#error: The #error preprocessor directive indicates error. The compiler gives fatal error if #error directive is found and skips further compilation process.

Code:

```
#include<stdio.h>
#ifndef __MATH_H
        #error First include then compile
#else
        void main(){
           float a;
           a=sqrt(7);
           printf("%f",a);
        }
#endif
```

Output:



```
bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_2$ cpp error.c >
 error.i
error.c:3:3: error: #error First include then compile
  #error First include then compile
   ^
```

Conclusion: From this practical I have learnt about differnt types of preprocessor directives and how all are works.