CHAP 6

THE C PREPROCESSOR

1. If a file to be included doesn't exist, the preprocessor flashes an error message.

Ans: TRUE.

2. The preprocessor can trap simple errors like missing declarations, nested comments or mismatch of braces.

Ans: FALSE.

3. O/p?

```
#define SQR(x) (x * x)
 main()
 {
  int a,b=3;
  a=SQR(b+2);
  printf("\n%d",a);
 }
```

a. 25
b. 11
c. error
d. garbage value

ans:B. Because on preprocessing the expression becomes (3+2 * 2+3).

4.How would you define SQR macro above in Q3 such as it gives the result of a as 25

ans: #SQR(x) ((x) * (x))

5. O/p?

```
#define CUBE(x) (x * x * x)
 main()
{
 int a,b=3;
 a=CUBE(b++);
 printf("\n %d %d",a,b);
 }
```

Ans: 27 6. Though some compilers may give this as an answer, according to ANSI C the expression b++ * b++ * b++ is undefined. Refer to chap 3 for more details.

6.Indicate what the SWAP macro be expanded to on preprocessing. Would the code compile?

```
#define SWAP(a,b,c) (c t; t=a, a=b, b=t;)
main()
{
 int x=10,y=20;
 SWAP(x,y,int);
 Printf("%d %d",x,y);
}
```

Ans: (int t; t=a, a=b, b=t;);
This code won't compile since the declaration of t can't take place within parenthesis.

7. How should you modify SWAP macro such that it can swap two integers?

Ans: #define SWAP(a,b,c) c t; t=a, a=b, b=t;

8.What is the limitation of SWAP macro above In Q7?

Ans: It can't swap pointer's for example the following code won't compile:-

```
#define SWAP(a,b,c) c t; t=a, a=b, b=t;
main()
{
```

```
 float x=10,y=20;
 float p,q;
 p=&x;  q=&y;
 SWAP(p, q, float);
 Printf("%f %f", x, y);
}
```

9.In which line of the following program, the error would be reported?

```
1   #define CIRCUM® (3.14 * r* r);
2   main()
3   {
4     float r=1.0.c;
5     c=CIRCUM(r);
6     printf("\n %f",c);
7     if(CIRCUM(r) == 6.28)
8        printf("\n Good Day!");
9    }
```

Ans: Line number 7. Whereas the culprit is really the semicolon in line 1.)n expansion line no 7 becomes if((3.14 * 1.0 *1.0);== 6.28). Hence the error.

10.What is the type of variable b in the following declaration?

```
#define FLOATPTR float *
FLOATPTR a,b;
```

Ans: FLOAT. Since on expansion declaration becomes float *a,b;

11.Is it necessary that the header files should have a .h extension?

Ans: NO.

12. What do the header files usually contain?

Ans: Preprocessor directives like #define, structure,union, and enum declarations, typedef declarations, global variable declarations and external function declarations. You should not write the actual code (function bodies) or global

variable definition (that is defining or initializing instances) in header files. The #include directive should be used to pull in header files, not other .c files.

13.Would it result into an error if a header file is included twice?

Ans: YES. Unless the header file has taken care to ensure that if already included it doesn't get included again.

14.How can a header file ensure that it doesn't get included more than once?

Ans: All declarations must be written in a manner shown bellow. Assume that the name of the header file is FUNCS.H.

```
/* FUNCS.H */
#ifndef _FUNCS
  #define _FUNCS
    /* all declarations would go here */
#endif
```

Now if we include this file twice as shown bellow, it would get included only once

15.On inclusion, where are the header files searched for?

Ans: If included using < > the files get searched in the predefined (can be changed) include path. If included using " " syntax in addition to the predefined include path the files also get searched in the current directory (usually the directory from which you invoke the comlier).

16.Would the following #typedef work?
typedef #include l;

Ans: NO. Because typedef goes to work after preprocessing.

17. Would the following code compile correctily?
main()

```
{
 #ifdef NOTE
    /* unterminated comment
    int a;
    a=10;
#else
  int a;
  a=20;
#endif

 printf("%d",a);
}
```

Ans: NO. Even though #ifdef fails in this case (NOTE being undefined) and the if block doesn't go for compilation errors in it are not permitted.

18. O/p?

```
#define MESS junk
main()
{
  printf("MESS");
}
```

Ans: MESS

19. Will the following program print the message infinite times? <Yes/No>

```
#define INFINITELOOP while(1)
main()
{
  INFINITELOOP
    Printf("\n HELLOW");
}
```

Ans

Ans: :YES.

20.O/p?

```
#define MAX(a,b) (a>b?a:b)
main()
{
  int x;
  x=MAX(3+2,2+7);
  printf("%d",x);
}
```

Ans: 9

21. O/p?

```
#define PRINT(int) printf("%d",int)
main()
{
 int x=2,y=3,z=4;
 PRINT(x);
 PRINT(y);
 PRINT(z);
}
```

Ans: 2 3 4

22.O/P?

```
#define PRINT(int) printf("int=%d",int)
main()
{
  int x=2,y=3,z=4;
  PRINT(x);
  PRINT(y);
  PRINT(z);
}
```

Ans: int=2 int=3 int=4

23.How would modify the macro of Q22 above such that it outputs
x=2 y=3 z=4

Ans:

```
#define PRINT(int) printf(#int"=%d",int)
main()
{
  int x=2, y=3, z=4;
 PRINT(x);
 PRINT(y);
 PRINT(z);
}
```

The rule is if the parameter name is preceded by a # in the macro expansion, the combination (of # and parameter) will be expanded into a quoted string with the parameter replaced by the actual argument. This can be combined with string concatenation to print the output desired in our program. On expansion the macro becomes

```
Printf("x" "=%d",x);
```

The two strings get concatenated, so the effect is

```
Printf("x=%d",x);
```

24.Would the following program compile successively? <Yes/No>

```
main()
{
   printf("Tips" "Traps");
}
```

Ans: Yes. The o/p is TipsTraps.

25. Define the macro DEBUG such that the following program outputs:

```
DEBUG: x=4;
DEBUG: y=3.140000
DEBUG: ch=A
```

```
main()
{
  int x=4;
  float y=3.14;
  char ch='A';
  DEBUG (x,"%d");
  DEBUG (a,"%f");
  DEBUG (ch,"%c");
}
```

Ans: #define DEBUG(var, Fmt) printf("DEBUG:" #var "=" #fmt "\n",ar)

26.O/p?

```
#define str(x) #x
#define Xstr(x) str(x)
#define oper multiply
main()
{
  char *opername=Xstr(oper);
  printf("%s",opername);
}
```

Ans: multiply

Here the two operations are being carried out expansion and stringizing. Xstr() macro expands its argument, and then str() stringizes it.

27. Write the macro PRINT for the following program such that it outputs:

```
x=4 y=4 z=5
a=1 b=2 c=3
main()
{
 int x=4, y=4, z=5;
 int a=1, b=2, c=3;
 PRINT(x,y,z);
 PRINT(a,b,c);
}
```

Ans: #define PRINT(var1, var2, var3) printf("\n" #var1 "=%d"#var2 "=%d" #var3 "=%d", var1,var2,var3)