Give the output of the programs in each case unless mentioned otherwise

1.
```
void main()
{
int d=5;
printf("%f",d);
}
```

Ans: Undefined

2.
```
void main()
{
int i;
for(i=1;i<4,i++)
switch(i)
case 1: printf("%d",i);break;
{
case 2:printf("%d",i);break;
case 3:printf("%d",i);break;
}
switch(i) case 4:printf("%d",i);
}
```

Ans: 1,2,3,4

3.
```
void main()
{
char *s="\12345s\n";
printf("%d",sizeof(s));
}
```

Ans: 6

4.
```
void main()
{
unsigned i=1; /* unsigned char k= -1 => k=255; */
signed j=-1; /* char k= -1 => k=65535 */
/* unsigned or signed int k= -1 =>k=65535 */
if(i<j)
printf("less");
else
if(i>j)
printf("greater");
else
if(i==j)
printf("equal");
}
```

Ans: less

5.

```
void main()
{
float j;
j=1000*1000;
printf("%f",j);
}
```

1. 1000000
2. Overflow
3. Error
4. None

Ans: 4

6. How do you declare an array of N pointers to functions returning pointers to functions returning pointers to characters?

Ans: The first part of this question can be answered in at least three ways:

1. char *(*(*a[N])())();

2. Build the declaration up incrementally, using typedefs:

```
typedef char *pc;    /* pointer to char */
typedef pc fpc();    /* function returning pointer to char */
typedef fpc *pfpc;    /* pointer to above */
typedef pfpc fpfpc();    /* function returning... */
typedef fpfpc *pfpfpc;    /* pointer to... */
pfpfpc a[N];         /* array of... */
```

3. Use the cdecl program, which turns English into C and vice versa:

```
cdecl> declare a as array of pointer to function returning
    pointer to function returning pointer to char
char *(*(*a[])())()
```

cdecl can also explain complicated declarations, help with casts, and indicate which set of parentheses the arguments go in (for complicated function definitions, like the one above).
Any good book on C should explain how to read these complicated C declarations "inside out" to understand them ("declaration mimics use").

The pointer-to-function declarations in the examples above have
not included parameter type information. When the parameters
have complicated types, declarations can *really* get messy.
(Modern versions of cdecl can help here, too.)

7. A structure pointer is defined of the type time . With 3 fields min,sec hours having pointers to intergers.
Write the way to initialize the 2nd element to 10.

8. In the above question an array of pointers is declared.
Write the statement to initialize the 3rd element of the 2 element to 10;

9.
```
int f()
void main()
{
f(1);
f(1,2);
f(1,2,3);
}
f(int i,int j,int k)
{
printf("%d %d %d",i,j,k);
}
```

What are the number of syntax errors in the above?

Ans: None.

10.
```
void main()
{
int i=7;
printf("%d",i++*i++);
}
```

Ans: 56

11.
```
#define one 0
#ifdef one
printf("one is defined ");
#ifndef one
printf("one is not defined ");
```

Ans: "one is defined"

12.
```
void main()
{
intcount=10,*temp,sum=0;
temp=&count;
*temp=20;
temp=&sum;
*temp=count;
printf("%d %d %d ",count,*temp,sum);
}
```

Ans: 20 20 20

13. There was question in c working only on unix machine with pattern matching.

14. what is alloca()

Ans : It allocates and frees memory after use/after getting out of scope

15.
```
main()
{
static i=3;
printf("%d",i--);
return i>0 ? main():0;
}
```

Ans: 321

16.
```
char *foo()
{
char result[100]);
strcpy(result,"anything is good");
return(result);
}
void main()
{
char *j;
j=foo()
printf("%s",j);
}
```

Ans: anything is good.

17.
```
void main()
{
```

```
char *s[]={ "dharma","hewlett-packard","siemens","ibm"};
char **p;
p=s;
printf("%s",++*p);
printf("%s",*p++);
printf("%s",++*p);
}
```

Ans: "harma" (p->add(dharma) && (*p)->harma)
"harma" (after printing, p->add(hewlett-packard) &&(*p)->harma)
"ewlett-packard"