```
1) struct cookie
2)   {
3)      char Name[30];
4)      float flour;
5)      int eggs;
6)      float sugar;
7)      float butter;
8)   };
```

Illustration 1:

What are the parts of a structure declaration:

1) In Illustration 1, which line uses the keyword?
   A) 8
   B) 2
   C) 1
   D) 3

2) In Illustration 1, what is the tag?
   A) cookie
   B) Name
   C) struct
   D) };

3) In Illustration 1, which line(s) form the body?
   A) 2-8
   B) 1
   C) 2
   D) 5-7

4) In Illustration 1, which choice is a data member?
   A) struct cookie
   B) sugar;
   C) };
   D) float flour;

5) What symbols define the body of a structure?
   A) Parenthesis ()
   B) Braces {}
   C) Bracket []
   D) Angle brackets <>

6) Which creates or instantiates the memory for a structure?
   A) Declaration
   B) Definition
   C) Delineation
   D) Dereference

7) Which operator (valid only in C++) allocates a new chunk of memory to hold a variable of type type and returns a pointer to that memory.
   A) const
   B) new
   C) struct
   D) data

8) How would an instance and a pointer for a cookie structure using the ***new*** operator?
   A) cookie chocolate = cookie new;
   B) cookie* chocolate = new cookie;
   C) cookie* new chocolate cookie;
   D) cookie chocolate = new cookie*;

9) Any object has 6 parts – what are they?
   A) container, data type, sign, value, address, pointer
   B) data, array, integer, scope, structure, pointer
   C) container, data type, name, value, address, scope
   D) data, data type, address, value, pointer, structure

10) In Illustration 1, line 1, structure is a
   A) Container
   B) Variable
   C) value
   D) address

11) In Illustration 1, How many variable members does the structure have?
   A) 1
   B) 2
   C) 6
   D) 4

12) In Illustration 1, How many array members does the structure have?
   A) 6
   B) 2
   C) 4
   D) 1

13) How many structure members can a structure like in Illustration 1 have?
   A) 0
   B) 1
   C) unlimited
   D) 6

14) How are structure members **directly** dereferenced?
```
A) y = cookie.sugar;
B) y = *cookie.sugar;
C) y = cookie_sugar;
D) y = cookie(sugar);
```

15) How are structure members indirectly dereferenced
```
A) y = cookie>>chip;
B) y = *cookie->chip;
C) y = cookie->chip;
D) y = cookie>(chip);
```

16) What type of object reference is *car* in car->door?
A) direct pointer
B) direct reference
C) pointer to direct
D) structure pointer

17) What type of instance is the object *car* in car->door?
A) Named block of memory
B) Unnamed block of memory with named pointer
C) Unnamed pointer to block of memory
D) Unnamed block of memory

18) What type of object reference is *car* in car.door?
A) structure pointer
B) direct reference
C) pointer to direct
D) direct pointer

19) What type of instance is the object *car* in car.door?
A) Unnamed block of memory with named pointer
B) Unnamed pointer to block of memory
C) Named block of memory
D) Unnamed block of memory

20) What is *door* in car->door?
A) Pointer
B) member
C) direct
D) structure

21) In reference to Illustration 2 How would a function foo be declared with a dollar structure as a calling argument?
```
A) int foo (struct* dollar x);
B) int foo (struct dollar x);
C) int foo (struct dollar);
D) int foo (int dollar x);
```

22) In reference to Illustration 2 How would a dollar named '**MyMoney**' be declared directly?
```
A) struct dollar MyMoney;
B) struct * dollar MyMoney;
C) struct MyMoney;
D) struct MyMoney dollar;
```

```
1) struct dollar
2) {
3)     int quarter;
4)     int dime;
5)     int nickel;
6)     int penny;
7) };
```

Illustration 2:

23) In reference to Illustration 2 How would *foo* be called with **MyMoney** as a calling argument?
```
A) z = foo( int MyMoney );
B) z = foo( struct MyMoney );
C) z = foo( struct dollar);
D) z = foo( MyMoney );
```

24) In reference to Illustration 2 How would one of the components of a dollar be printed in function *foo*?
```
A) printf("\nThe change had %d dimes\n",dime.x);
B) printf("\nThe change had %d dimes\n",x->dime);
C) printf("\nThe change had %s dimes\n",x.dime);
D) printf("\nThe change had %d dimes\n",x.dime);
```

25) In reference to Illustration 2 How could the parts of a dollar be initialized as part of the declaration?
```
A) struct dollar Mine={3, 2, 1, 0,};
B) struct dollar Mine={3; 2; 1; 0};
C) struct dollar Mine={3, 2, 1, 0};
D) struct dollar Mine=[3, 2, 1, 0];
```

26) In reference to Illustration 2 How could another dollar instance be created and the pointer '**MyMoneyPtr**' be set to the dollar location?
```
A) new dollar* MyMoneyPtr = dollar;
B) dollar MyMoneyPtr = new dollar*;
C) dollar* MyMoneyPtr = new dollar;
D) new dollar MyMoneyPtr = dollar*;
```

27) In reference to Illustration 2 How would a function *foo2* be declared with a dollar pointer as a calling argument?
```
A) int foo2 (struct dollar* xPtr);
B) int foo2 (struct* dollar xPtr);
C) int foo2 (struct dollar xPtr*);
D) int foo2 (*struct dollar xPtr);
```

28) In reference to Illustration 2 How could *foo2* be called with **MyMoneyPtr**?
```
A) z = foo2( MyMoneyPtr* );
B) z = foo2( struct MyMoneyPtr );
C) z = foo2( dollar* MyMoneyPtr );
D) z = foo2( MyMoneyPtr );
```

29) In reference to Illustration 2 How could *foo2* print one part of a dollar as the calling argument?
```
A) printf("\nThe change had %d dimes\n",x.dime);
B) printf("\nThe change had %d dimes\n",x->dime);
C) printf("\nThe change had %s dimes\n",x.dime);
D) printf("\nThe change had %d dimes\n",dime.x);
```

30) In reference to Illustration 2 Could foo2 change the contents of the parent dollar?
A) Yes
B) no
C) indirectly
D) using recursion

31) In reference to Illustration 2 Could foo change the contents of the parent dollar?
A) Yes
B) using recursion
C) indirectly
D) No

File I/O: binary
1. Create a FILE pointer **myfunds**
2. Open a ***file*** for binary write
3. Binary write block of memory containing **money** array
4. Close the ***file***
5. Open ***file*** for binary read
6. Binary read into block of memory containing **savings**

Illustration 3:

32) In Illustration 3-1When declaring a file pointer, *FILE* is used. What type of object is *FILE*?
A) a predefined system operator
B) a predefined system pointer
C) a predefined system function
D) a predefined system structure

33) In Illustration 3-1 How would a file pointer named **myfunds** be declared?
```
A) FILE  myfunds;
B) FILE * myfunds;
C) FILE [myfunds];
D) myfunds FILE *;
```

34) In Illustration 3-2What statement would correctly open the file "funds.val" and initialize the file pointer to write binary information?
```
A) fopen("myfunds", "wb");
B) myfunds = fopen("funds.val");
C) myfunds = fopen("funds.val", "w");
D) myfunds = fopen("funds.val", "wb");
```

35) In Illustration 3-2What happens if fopen returns NULL?
A) void, so no return argument used
B) file is successfully opened
C) file could not be opened
D) continue to use NULL pointer

36) In Illustration 3-3 What statement would write
int money[30];
to "funds.val" file?
```
A) fwrite(money, sizeof(int), myfunds);
B) fwrite(money, 30, myfunds);
C) fwrite(money, sizeof(int), 30, myfunds);
D) fwrite(myfunds, sizeof(int), 30, money)
```

37) In Illustration 3-4 What statement would close "funds.val"?
```
A) fclose (myfunds);*
B) fclose(money);
C) FileClose(money, myfunds);
D) FILE("close", myfunds ,money);
```

38) In Illustration 3-5 What statement would open "funds.val" file to read binary?
```
A) myfunds = fopen("funds.val", "rb");
B) FILE fopen("funds.val", "r", myfunds);
C) myfunds = fopen("funds.val", "b");
D) fopen(myfunds, "b", "funds.val");
```

39) In Illustration 3-6 What statement would read
int savings[30];
from "funds.val" file?
```
A) nread = fread(fp, savings);
B) nread = fread(fp, sizeof(int), 30, savings);
C) nread = fread(savings, fp);
D) nread = fread(savings, sizeof(int), 30, fp);
```

40) In Illustration 3-6 What value does **nread** get after reading
int savings[30];
from "funds.val" file?
```
   nread = fread(savings, sizeof(int), 30, fp);

A) 120 if successful
B) 0 if successful
C) 30 if successful
D) 1 if successful
```

File I/O: Strings
1. Create string **str** initialized to alphabet
2. Create a FILE pointer **MyTxt**
3. Open ***file*** "message.txt" for text write
4. Write string to ***file***
5. Close the ***file***
6. Open ***file*** for text read
7. Read string from ***file***
8. Rewind to start of ***file***
9. Read 5 characters from ***file***
10. Position to letter 'j' in ***file***
11. Read 5 characters starting at 'j' in **file**

Illustration 4:

41) In Illustration 4-1 What statement would create a string called *str* initialized to lower case alphabet?
```
A) char str[] = "abcdefghijklmnopqrstuvwxyz";
B) char str[] = {"abcdefghijklmnopqrstuvwxyz"};
C) char* str[] = "abcdefghijklmnopqrstuvwxyz";
D) char str[] = [abcdefghijklmnopqrstuvwxyz];
```

42) In Illustration 4-2 What statement would create a file pointer **MyTxt**?
```
A)   MyTxt FILE*;
B)   FILE MyTxt;
C)   FILE* MyTxt;
D)   MyTxt* FILE;
```

43) In Illustration 4-3 What statement would open the file "message.txt" to write text with **MyTxt**?
```
A)   MyTxt = fopen("message.txt", "wb");
B)   MyTxt = fopen("message.txt", "w");
C)   MyTxt = fopen("w", "message.txt");
D)   MyTxt = fopen("message.txt");
```

44) In Illustration 4-4 What statement would put **str** into "message.txt" file?
```
A) fputs(MyTxt);
B) fwrite(str, MyTxt);
C) fputs(MyTxt, str);
D) fputs(str, MyTxt);
```

45) In Illustration 4-4 What alternative statement would put *str* into "message.txt" file?
```
A) fprintf(str);
B) fprintf(MyTxt,str);
C) fprintf(MyTxt,"%s");
D) fprintf(MyTxt,"%s",str);
```

46) In Illustration 4-5What statement would close "message.txt" file?
```
A) fclose (MyTxt);
B) fclose(str);
C) FileClose("message.txt");
D) FILE(close, MyTxt);
```

47) In Illustration 4-6 What statement would open the file "message.txt" to read text with **MyTxt**?
```
A) MyTxt = fopen("message.txt");
B) MyTxt = fopen("message.txt", "rb");
C) MyTxt = fopen("message.txt", "r");
D) MyTxt = fopen( "r", "message.txt");
```

48) In Illustration 4-7What statement would read **int buf[80]**; from "message.txt" file?
```
A) fgets(buf, 80, MyTxt);
B) finput(buf, 80, MyTxt);
C) fgets(buf, MyTxt);
D) finput(80, MyTxt);
```

49) In Illustration 4-7 If the alphabet from "message.txt" file is read into **buf**, how many characters will be read?
A) 26 letters
B) 27 (26 letters + '\0')
C) 28 (26 letters + '\0')
D) 80

50) In Illustration 4-8 How can "message.txt" file rewind to the beginning?
```
A) rewind("message.txt");
B) rewind(MyTxt);
C) rewind(start);
D) rewind(80, MyTxt);
```

51) In Illustration 4-9 How can only 5 characters be read from "message.txt" file after rewind?
```
A) fgets(buf);
B) fgets(buf, 5, MyTxt);
C) fgets(buf, 6, MyTxt);
D) fgets(MyTxt, 5);
```

52) In Illustration 4-10 How can the file pointer be positioned just before the letter ***j*** in "message.txt" file?
```
A) fseek(fp, 9, SEEK_CUR);
B) fseek(fp, 9, SEEK_SET);
C) fseek(fp, 9, SEEK_END);
D) fseek(fp, 9);
```

53) In Illustration 4-10 How can 5 letters including the letter ***j*** be read from "message.txt" file after positioning the file pointer just before the letter ***j***?
```
A) fgets(buf);
B) fgets(buf, 5, MyTxt);
C) fgets(buf, 6, MyTxt);
D) fgets(MyTxt, 5);
```

```
1)  #include <stdio.h>
2)  #define BUFSIZE 100
3)  main()
4)  {
5)      char buf[BUFSIZE];
6)      char filename[20];
7)      FILE *fp;
8)      puts("Enter text file to open: ");
9)      gets(filename);
10)     if ((fp = fopen(filename, "r"))==NULL)
11)     {
12)      fprintf(stderr, "Error opening file.");
13)      return(1);
14)     }
15)
16)     while ( !feof(fp) )
17)     {
18)         fgets(buf, BUFSIZE, fp);
19)         printf("%s",buf);
20)     }
21)     fclose(fp);
22) }
```
Illustration 5:

54) In Illustration 5 which line declares a file pointer?
A) 18
B) 1
C) 10
D) 7

55) In Illustration 5 which line opens a file for text input?
A) 7
B) 18
C) 13
D) 10

56) In Illustration 5 which line outputs to **stdout**
A) 9
B) 10
C) 18
D) 19

57) In Illustration 5 which line 16 uses feof that returns what if the input file has been completely read in?
A) Yes
B) False
C) True
D) No

58) In Illustration 5 which line gets a file name from **stdin**?
A) 6
B) 18
C) 9
D) 8

59) In Illustration 5 in line 21, **fclose** calling argument is what?
A) pointer
B) value
C) file name
D) structure

60) In Illustration 5 which line closes **stdout**?
A) 21
B) none
C) 19
D) 7