**1**. C99 standard guarantees uniqueness of \_\_\_\_\_\_\_\_\_\_ characters for internal names.  
a) 31  
b) 63  
c) 12  
d) 14  
Answer: b  
Explanation: ISO C99 compiler may consider only first 63 characters for internal names.

**2**. C99 standard guarantees uniqueness of \_\_\_\_\_\_\_\_\_\_\_ characters for external names.  
a) 31  
b) 6  
c) 12  
d) 14  
Answer: a  
Explanation: ISO C99 compiler may consider only first 31 characters for external names.

**3**. Which of the following is not a valid variable name declaration?  
a) int \_\_a3;  
b) int \_\_3a;  
c) int \_\_A3;  
d) None of the mentioned  
Answer: d

Explanation: Variable can start with underscore(\_).  
**4**. Which of the following is not a valid variable name declaration?  
a) int \_a3;  
b) int a\_3;  
c) int 3\_a;  
d) int \_3a  
Answer: c  
Explanation: Variable name cannot start with a digit.

**5**. All keywords in C are in \_\_\_\_\_\_\_\_\_\_\_\_  
a) LowerCase letters  
b) UpperCase letters  
c) CamelCase letters  
d) None of the mentioned  
Answer: a  
Explanation:Some keywords like: case, auto, register, default etc all keywords are lowercase.

**6.** Variable name resolution (number of significant characters for the uniqueness of variable) depends on \_\_\_\_\_\_\_\_\_\_\_  
a) Compiler and linker implementations  
b) Assemblers and loaders implementations  
c) C language  
d) None of the mentioned  
Answer: a  
Explanation: It depends on the standard to which compiler and linkers are adhering.

**7.** Which of the following is not a valid C variable name?  
a) int number;  
b) float rate;  
c) int variable\_count;  
d) int $main;

Answer: d  
Explanation: Since only underscore and no other special character is allowed in a variable name, it results in an error.

**8**. Which of the following is true for variable names in C?  
a) They can contain alphanumeric characters as well as special characters  
b) It is not an error to declare a variable to be one of the keywords(like goto, static)  
c) Variable names cannot start with a digit  
d) Variable can be of any length  
Answer: c  
Explanation: According to the syntax for C variable name, it cannot start with a digit

**9.** Which is valid C expression?  
a) int my\_num = 100,000;  
b) int my\_num = 100000;  
c) int my num = 1000;  
d) int $my\_num = 10000;  
Answer: b  
Explanation: Space, comma and $ cannot be used in a variable name.

**10**.What will be the output of the following C code?

#include <stdio.h>

int main()

{

printf("Hello World! %d **\n**", x);

return 0;

}

a) Hello World! x;  
b) Hello World! followed by a junk value  
c) Compile time error  
d) Hello World!  
Answer: c  
Explanation: It results in an error since x is used without declaring the variable x.  
Output:  
$ cc pgm1.c  
pgm1.c: In function ‘main’:  
pgm1.c:4: error: ‘x’ undeclared (first use in this function)  
pgm1.c:4: error: (Each undeclared identifier is reported only once  
pgm1.c:4: error: for each function it appears in.)

**11**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

int y = 10000;

int y = 34;

printf("Hello World! %d**\n**", y);

return 0;

}

a) Compile time error  
b) Hello World! 34  
c) Hello World! 1000  
d) Hello World! followed by a junk value

Answer: a  
Explanation: Since y is already defined, redefining it results in an error.  
Output:  
$ cc pgm2.c  
pgm2.c: In function ‘main’:  
pgm2.c:5: error: redefinition of ‘y’  
pgm2.c:4: note: previous definition of ‘y’ was here

**12.** Which of the following is not a valid variable name declaration?  
a) float PI = 3.14;  
b) double PI = 3.14;  
c) int PI = 3.14;  
d) #define PI 3.14  
Answer: d  
Explanation: #define PI 3.14 is a macro preprocessor, it is a textual substitution.

**13**. What will happen if the following C code is executed?

#include <stdio.h>

int main()

{

int main = 3;

printf("%d", main);

return 0;

}

a) It will cause a compile-time error  
b) It will cause a run-time error  
c) It will run without any error and prints 3  
d) It will experience infinite looping

Answer: c  
Explanation: A C program can have same function name and same variable name.  
**14**. What is the problem in the following variable declaration?

float 3Bedroom-Hall-Kitchen?;

a) The variable name begins with an integer  
b) The special character ‘-‘  
c) The special character ‘?’  
d) All of the mentioned

Answer: d  
Explanation: A variable name cannot start with an integer, along with that the C compiler interprets the ‘-‘ and ‘?’ as a minus operator and a question mark operator respectively.

**15**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

int ThisIsVariableName = 12;

int ThisIsVariablename = 14;

printf("%d", ThisIsVariablename);

return 0;

}

a) The program will print 12  
b) The program will print 14  
c) The program will have a runtime error  
d) The program will cause a compile-time error due to redeclaration  
Answer: b  
Explanation: Variable names ThisIsVariablename and ThisIsVariableName are both distinct as C is case sensitive.  
**16**. Which of the following cannot be a variable name in C?  
a) volatile  
b) true  
c) friend  
d) export  
Answer: a  
Explanation: volatile is C keyword.

**17**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

int a[5] = {1, 2, 3, 4, 5};

int i;

for (i = 0; i < 5; i++)

if ((char)a[i] == '5')

printf("%d**\n**", a[i]);

else

printf("FAIL**\n**");

}

a) The compiler will flag an error  
b) The program will compile and print the output 5  
c) The program will compile and print the ASCII value of 5  
d) The program will compile and print FAIL for 5 times  
Answer: d  
Explanation: The ASCII value of 5 is 53, the char type-casted integral value 5 is 5 only.  
Output:  
$ cc pgm1.c  
$ a.out  
FAIL  
FAIL  
FAIL  
FAIL  
FAIL

**18**. The format identifier ‘%i’ is also used for \_\_\_\_\_ data type.  
a) char  
b) int  
c) float  
d) double  
Answer: b  
Explanation: Both %d and %i can be used as a format identifier for int data type.

**19**. Which data type is most suitable for storing a number 65000 in a 32-bit system?  
a) signed short  
b) unsigned short  
c) long  
d) int  
Answer: b  
Explanation: 65000 comes in the range of short (16-bit) which occupies the least memory. Signed short ranges from -32768 to 32767 and hence we should use unsigned short.

**20**. Which of the following is a User-defined data type?  
a) typedef int Boolean;  
b) typedef enum {Mon, Tue, Wed, Thu, Fri} Workdays;  
c) struct {char name[10], int age};  
d) all of the mentioned  
Answer: d  
Explanation: typedef and struct are used to define user-defined data types.

**21**. What is the size of an int data type?  
a) 4 Bytes  
b) 8 Bytes  
c) Depends on the system/compiler  
d) Cannot be determined  
Answer: c  
Explanation: The size of the data types depend on the system.

**22.** What will be the output of the following C code?

#include <stdio.h>

int main()

{

signed char chr;

chr = 128;

printf("%d**\n**", chr);

return 0;

}

a) 128  
b) -128  
c) Depends on the compiler  
d) None of the mentioned

Answer: b  
Explanation: signed char will be a negative number.

**23**. What is short int in C programming?  
a) The basic data type of C  
b) Qualifier  
c) Short is the qualifier and int is the basic data type  
d) All of the mentioned  
Answer: c

**24**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

float f1 = 0.1;

if (f1 == 0.1)

printf("equal**\n**");

else

printf("not equal**\n**");

}

a) equal  
b) not equal  
c) output depends on the compiler  
d) error  
Answer: b  
Explanation: 0.1 by default is of type double which has different representation than float resulting in inequality even after conversion.

**25**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

float f1 = 0.1;

if (f1 == 0.1f)

printf("equal**\n**");

else

printf("not equal**\n**");

}

a) equal  
b) not equal  
c) output depends on compiler  
d) error  
Answer: a  
Explanation: 0.1f results in 0.1 to be stored in floating point representations.  
**26**. What will be the output of the following C code on a 32-bit machine?

#include <stdio.h>

int main()

{

int x = 10000;

double y = 56;

int \*p = &x;

double \*q = &y;

printf("p and q are %d and %d", sizeof(p), sizeof(q));

return 0;

}

a) p and q are 4 and 4  
b) p and q are 4 and 8  
c) compiler error  
d) p and q are 2 and 8  
Answer: a  
Explanation: Size of any type of pointer is 4 on a 32-bit machine.  
**27**. Which is correct with respect to the size of the data types?  
a) char > int > float  
b) int > char > float  
c) char < int < double  
d) double > char > int

Answer: c  
Explanation: char has less bytes than int and int has less bytes than double in any system

**28**. What will be the output of the following C code on a 64 bit machine?

#include <stdio.h>

union Sti

{

int nu;

char m;

};

int main()

{

union Sti s;

printf("%d", sizeof(s));

return 0;

}

a) 8  
b) 5  
c) 9  
d) 4  
Answer: d  
Explanation: Since the size of a union is the size of its maximum data type, here int is the largest data type. Hence the size of the union is 4.  
**29**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

float x = 'a';

printf("%f", x);

return 0;

}

a) a  
b) run time error  
c) a.0000000  
d) 97.000000  
Answer: d  
Explanation: Since the ASCII value of a is 97, the same is assigned to the float variable and printed.

**30**. Which of the data types has the size that is variable?  
a) int  
b) struct  
c) float  
d) double

Answer: b  
Explanation: Since the size of the structure depends on its fields, it has a variable size.

**31**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

**enum** {ORANGE = 5, MANGO, BANANA = 4, PEACH};

printf("PEACH = %d**\n**", PEACH);

}

a) PEACH = 3  
b) PEACH = 4  
c) PEACH = 5  
d) PEACH = 6

Answer: c  
Explanation: In enum, the value of constant is defined to the recent assignment from left.

**32**. In the following code snippet, character pointer str holds a reference to the string \_\_\_\_\_\_\_\_\_\_\_

char \*str = "Sanfoundry.com\0" "training classes";

a) Sanfoundry.com  
b) Sanfoundry.com\0training classes  
c) Sanfoundry.comtraining classes  
d) Invalid declaration

Answer: b  
Explanation: ‘\0’ is accepted as a char in the string. Even though strlen will give length of string “Sanfoundry.com”, in memory str is pointing to entire string including training classes.

**33**. What will be the output of the following C code?

#include <stdio.h>

#define a 10

int main()

{

const int a = 5;

printf("a = %d**\n**", a);

}

a) a = 5  
b) a = 10  
c) Compilation error  
d) Runtime error

Answer: c  
Explanation: The #define substitutes a with 10 without leaving any identifier, which results in Compilation error.  
**34**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

int var = 010;

printf("%d", var);

}

a) 2  
b) 8  
c) 9  
d) 10  
Answer: b  
Explanation:0 starts means octal. 010 is octal representation of 8.

**35**. What will be the output of the following C function?

#include <stdio.h>

**enum** birds {SPARROW, PEACOCK, PARROT};

**enum** animals {TIGER = 8, LION, RABBIT, ZEBRA};

int main()

{

**enum** birds m = TIGER;

int k;

k = m;

printf("%d**\n**", k);

return 0;

}

a) 0  
b) Compile time error  
c) 1  
d) 8  
Answer: d

**36**. What will be the output of the following C code?

#include <stdio.h>

#define MAX 2

**enum** bird {SPARROW = MAX + 1, PARROT = SPARROW + MAX};

int main()

{

**enum** bird b = PARROT;

printf("%d**\n**", b);

return 0;

}

a) Compilation error  
b) 5  
c) Undefined value  
d) 2

Answer: b  
Explanation: MAX value is 2 and hence PARROT will have value 3 + 2.

**37**. What will be the output of the following C code?

#include <stdio.h>

#include <string.h>

int main()

{

char \*str = "x";

char c = 'x';

char ary[1];

ary[0] = c;

printf("%d %d", strlen(str), strlen(ary));

return 0;

}

a) 1 1  
b) 2 1  
c) 2 2  
d) 1 (undefined value)

Answer: d  
Explanation: str is null terminated, but ary is not null terminated.  
Output:  
1 5

**38**. What will be the output of the following C code?

#include <stdio.h>

void foo(const int \*);

int main()

{

const int i = 10;

printf("%d ", i);

foo(&i);

printf("%d", i);

}

void foo(const int \*i)

{

\*i = 20;

}

a) Compile time error  
b) 10 20  
c) Undefined value  
d) 10

Answer: a  
Explanation: Cannot change a const type value.

**39**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

const int i = 10;

int \*ptr = &i;

\*ptr = 20;

printf("%d**\n**", i);

return 0;

}

a) Compile time error  
b) Compile time warning and printf displays 20  
c) Undefined behavior  
d) 10

Answer: b  
Explanation: Changing const variable through non-constant pointers invokes compiler warning.  
**40**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

j = 10;

printf("%d**\n**", j++);

return 0;

}

a) 10  
b) 11  
c) Compile time error  
d) 0  
Answer: c  
Explanation: Variable j is not defined.

**41**. Will the following C code compile without any error?

#include <stdio.h>

int main()

{

for (int k = 0; k < 10; k++);

return 0;

}

a) Yes  
b) No  
c) Depends on the C standard implemented by compilers  
d) Error

Answer: c

**42**. Will the following C code compile without any error?

#include <stdio.h>

int main()

{

int k;

{

int k;

for (k = 0; k < 10; k++);

}

}

a) Yes  
b) No  
c) Depends on the compiler  
d) Depends on the C standard implemented by compilers

Answer: a  
Explanation: There can be blocks inside the block. But within a block, variables have only block scope.  
**43**. Which of the following declaration is not supported by C?  
a) String str;  
b) char \*str;  
c) float str = 3e2;  
d) Both String str; & float str = 3e2;

Answer: a  
Explanation: It is legal in Java, but not in C.

**44**. Which of the following format identifier can never be used for the variable var?

#include <stdio.h>

int main()

{

char \*var = "Advanced Training in C ";

}

a) %f  
b) %d  
c) %c  
d) %s  
Answer: a  
Explanation: %c can be used to print the indexed position.  
%d can still be used to display its ASCII value.  
%s is recommended.  
%f cannot be used for the variable var.

**45**.What will be the output of the following C code?

#include <stdio.h>

int main()

{

int i = -3;

int k = i % 2;

printf("%d**\n**", k);

}

a) Compile time error  
b) -1  
c) 1  
d) Implementation defined

Answer: b  
Explanation: here, -3%2=-1.5. As it is a int type data tai it takes -1.

**46**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

int i = 3;

int l = i / -2;

int k = i % -2;

printf("%d %d**\n**", l, k);

return 0;

}

a) Compile time error  
b) -1 1  
c) 1 -1  
d) Implementation defined

Answer: b

**47**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

int i = 5;

i = i / 3;

printf("%d**\n**", i);

return 0;

}

a) Implementation defined  
b) 1  
c) 3  
d) Compile time error

Answer: b

**48**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

int i = -5;

i = i / 3;

printf("%d**\n**", i);

return 0;

}

a) Implementation defined  
b) -1  
c) -3  
d) Compile time error

Answer: b

**49**. What will be the final value of x in the following C code?

#include <stdio.h>

void main()

{

int x = 5 \* 9 / 3 + 9;

}

a) 3.75  
b) Depends on compiler  
c) 24  
d) 3

Answer: c

**50**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

int a = 1, b = 1, c;

c = a++ + b;

printf("%d, %d", a, b);

}

a) a = 1, b = 1  
b) a = 2, b = 1  
c) a = 1, b = 2  
d) a = 2, b = 2

Answer: b  
Explanation: a=2 because a will be incremented.

**51**. For which of the following, “PI++;” code will fail?  
a) #define PI 3.14  
b) char \*PI = “A”;  
c) float PI = 3.14;  
d) none of the Mentioned

Answer: a

**52**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

int a = 10, b = 10;

if (a = 5)

b--;

printf("%d, %d", a, b--);

}

a) a = 10, b = 9  
b) a = 10, b = 8  
c) a = 5, b = 9  
d) a = 5, b = 8

Answer: c  
Explanation: a=5, here it is not condition. after this line a becomes 5. As it is true , b- - will be executed.

**53**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

int i = 0;

int j = i++ + i;

printf("%d**\n**", j);

}

a) 0  
b) 1  
c) 2  
d) Compile time error

Answer: b

**54**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

int i = 2;

int j = ++i + i;

printf("%d**\n**", j);

}

a) 6  
b) 5  
c) 4  
d) Compile time error

Answer: a

**55**. What will be the output of the following C code?

#include <stdio.h>

int x;

void main()

{

if (x)

printf("hi");

else

printf("how are u");

}

a) hi  
b) how are you  
c) compile time error  
d) error

Answer: b  
Explanation: As value of x is not defined , x value becomes 0 because it is int datatype. Condition will be false because x=0, so else will be execute.

**56**. What will be the output of the following C code?

#include <stdio.h>

void main()

{

int x = 0;

if (x == 0)

printf("hi");

else

printf("how are u");

printf("hello");

}

a) hi  
b) how are you  
c) hello  
d) hihello

Answer: d

**57**. What will be the output of the following C code?

#include <stdio.h>

void main()

{

int x = 5;

if (x < 1);

printf("Hello");

}

a) Nothing  
b) Run time error  
c) Hello  
d) Varies

Answer: c

**58**. What will be the output of the following C code? (Assuming that we have entered the value 1 in the standard input)

#include <stdio.h>

void main()

{

double ch;

printf("enter a value between 1 to 2:");

scanf("%lf", &ch);

switch (ch)

{

case 1:

printf("1");

**break**;

case 2:

printf("2");

**break**;

}

}

a) Compile time error  
b) 1  
c) 2  
d) Varies

Answer: a  
Explanation: Case read only integer and character data.

**59**. What will be the output of the following C code? (Assuming that we have entered the value 1 in the standard input)

#include <stdio.h>

void main()

{

int ch;

printf("enter a value between 1 to 2:");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("1**\n**");

default:

printf("2**\n**");

}

}

a) 1  
b) 2  
c) 1 2  
d) Run time error  
Answer: c  
**60**. What will be the output of the following C code? (Assuming that we have entered the value 2 in the standard input)

#include <stdio.h>

void main()

{

int ch;

printf("enter a value between 1 to 2:");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("1**\n**");

**break**;

printf("Hi");

default:

printf("2**\n**");

}

}

a) 1  
b) Hi 2  
c) Run time error  
d) 2

Answer: d

**61**. What will be the output of the following C code? (Assuming that we have entered the value 1 in the standard input)

#include <stdio.h>

void main()

{

int ch;

printf("enter a value between 1 to 2:");

scanf("%d", &ch);

switch (ch, ch + 1)

{

case 1:

printf("1**\n**");

**break**;

case 2:

printf("2");

**break**;

}

}

a) 1  
b) 2  
c) 3  
d) Run time error

Answer: b  
**62**. The C code ‘for(;;)’ represents an infinite loop. It can be terminated by \_\_\_\_\_\_\_\_\_\_\_  
a) break  
b) exit(0)  
c) abort()  
d) terminate

Answer: a  
Explanation: break keywords terminate the loop.

**63**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

short i;

for (i = 1; i >= 0; i++)

printf("%d**\n**", i);

}

a) The control won’t fall into the for loop  
b) Numbers will be displayed until the signed limit of short and throw a runtime error  
c) Numbers will be displayed until the signed limit of short and program will successfully terminate  
d) This program will get into an infinite loop and keep printing numbers with no errors

Answer: c

**64**. What will be the output of the following C code?

#include <stdio.h>

void main()

{

int k = 0;

for (k < 3; k++)

printf("Hello");

}

a) Compile time error  
b) Hello is printed thrice  
c) Nothing  
d) Varies

Answer: a  
Explanation: for( ; ;) loop should have this syntax.

**65**. What will be the output of the following C code?

#include <stdio.h>

void main()

{

double k = 0;

for (k = 0.0; k < 3.0; k++)

printf("Hello");

}

a) Run time error  
b) Hello is printed thrice  
c) Hello is printed twice  
d) Hello is printed infinitely

Answer: b

**66**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

int i = 0;

do {

i++;

printf("In while loop**\n**");

} while (i < 3);

}

a)

In while loop

In while loop

In while loop

b)

In while loop

In while loop

c) Depends on the compiler  
d) Compile time error

Answer: a

**68**.Which keyword can be used for coming out of recursion?  
a) break  
b) return  
c) exit  
d) both break and return

Answer: b

**69**. The keyword ‘break’ cannot be simply used within \_\_\_\_\_\_\_\_\_  
a) do-while  
b) if-else  
c) for  
d) while

Answer: b  
Explanation: break keyword usually used in loop to break the loop.

**70**. What will be the output of the following C code?

#include <stdio.h>

int main()

{

void foo();

void f()

{

foo();

}

f();

}

void foo()

{

printf("2 ");

}

a) 2 2  
b) 2  
c) Compile time error  
d) Depends on the compiler

Answer: d  
Explanation: Even though the answer is 2, this code will compile fine only with gcc. GNU C supports nesting of functions in C as a language extension whereas standard C compiler doesn’t.

**71**. What will be the output of the following C code?

#include <stdio.h>

void foo();

int main()

{

void foo(int);

foo(1);

return 0;

}

void foo(int i)

{

printf("2 ");

}

a) 2  
b) Compile time error  
c) Depends on the compiler  
d) Depends on the standard

Answer: a

**72**. There are two groups of string functions defined in the header <string.h>. What are they?  
a) first group names beginning with str; second group names beginning with mem  
b) first group names beginning with str; second group names beginning with is  
c) first group names beginning with string; second group names beginning with mem  
d) first group names beginning with str; second group names beginning with type

Answer: a  
Explanation: There are two groups of string functions declared under the header <string.h>. The first have names beginning with str and second have names beginning with mem.

**73**. What is the use of function char \*strchr(ch, c)?  
a) return pointer to first occurrence of ch in c or NULL if not present  
b) return pointer to first occurrence of c in ch or NULL if not present  
c) return pointer to first occurrence of ch in c or ignores if not present  
d) return pointer to first occurrence of cin ch or ignores if not present

Answer: b  
Explanation: The given code char \*strchr(ch, c) return pointer to first occurrence of c in ch or NULL if not present.

**74**. Which code from the given option return pointer to last occurrence of c in ch or NULL if not present?  
a) char \*strchr(ch, c)  
b) char \*strrchr(ch, c)  
c) char \*strncat(ch, c)  
d) char \*strcat(ch, c)

Answer: b  
Explanation: The function char \*strrchr(ch, c) returns pointer to last occurrence of c in ch or NULL if not present.

**75**. Which among the given options compares atmost n characters of string ch to string s?  
a) int strncmp(ch, s, n)  
b) int strcmp(ch, s)  
c) int strncmp(s, ch, n)  
d) int strcmp(s, ch)

Answer: a  
Explanation: int strncmp(ch, s, n) is used to compare at most n characters of string ch to string s; return <0 if ch0 of ch >s.

**76.**Which among the given options is the right explanation for the statement size\_t strcspn(c, s)?  
a) return length of prefix of s consisting of characters not in c  
b) return length of prefix of s consisting of characters present in c  
c) return length of prefix of c consisting of characters not in s  
d) return length of prefix of c consisting of characters present in s

Answer: c  
Explanation: The function size\_t strcspn(c, s) is used to return length of prefix of c consisting of characters not in s.

**77**. The mem functions are meant for \_\_\_\_\_\_\_  
a) returning a pointer to the token  
b) manipulating objects as character arrays  
c) returning a pointer for implemented-defined string  
d) returning a pointer to first occurrence of string in another string

Answer: b  
Explanation: The mem functions is used for manipulating objects as character arrays.

**78**. What is the function of void \*memset(s, c, n)?  
a) places character s into first n characters of c, return c  
b) places character c into first n characters of s, return s  
c) places character s into first n characters of c, return s  
d) places character c into first n character of s, return c

Answer: b  
Explanation: The void \*memset(s, c, n) places character c into first n characters of s, return s.

**79.** Functions whose names begin with “strn”  
a) manipulates sequences of arbitrary characters  
b) manipulates null-terminated sequences of characters  
c) manipulates sequence of non – null characters.  
d) returns a pointer to the token

Answer: c  
Explanation: Functions whose names begin with “strn” manipulates the sequence of non-null characters.

**80**. Which of the following is the right syntax to copy n characters from the object pointed to by s2 into the object pointed to by s1?  
a) void \*memcpy(void \*s1,const void \*s2,size\_t n);  
b) void \*memcpy(void \*s2, const void \*s1, size\_t n);  
c) void memcpy(void \*s1,const void \*s2, size\_t n);  
d) void memcpy(void \*s2,const void \*s1,size\_t n);

Answer: a  
Explanation: The memcpy() function copies n characters from the object pointed to by s2 into the object pointed to by s1. If copying takes place between objects that overlap, the behavior is undefined.

**81**. What does the following function returns void \*memmove(void \*s1,const void s2, size\_t n);?  
a) returns the value of s1  
b) returns the value of s2  
c) doesn’t return any value  
d) returns the value of s1 and s2

Answer: a  
Explanation: The memmove() function copies n characters from the object pointed to by s2 into the object pointed to by s1.The memmove() function returns the value of s1.

**82**. Which among the following is Copying function?  
a) memcpy()  
b) strcopy()  
c) memcopy()  
d) strxcpy()

Answer: a  
Explanation: The memcpy() function is used to copy n characters from the object.  
The code is void \*memcpy(void \*s1,const void \*s2, size\_t n).

**83**. Which function will you choose to join two words?  
a) strcpy()  
b) strcat()  
c) strncon()  
d) memcon()

Answer: b  
Explanation: The strcat() function is used for concatenating two strings, appends a copy of the string.  
char \*strcat(char \*s1,const char \*s2);

**84**. The \_\_\_\_\_\_ function appends not more than n characters.  
a) strcat()  
b) strcon()  
c) strncat()  
d) memcat()

Answer: c  
Explanation: The strncat() function appends not more than n characters from the array(s2) to the end of the string(s1).char \*strncat(char \*s1, const char \*s2,size\_t n);

**85**. What will strcmp() function do?  
a) compares the first n characters of the object  
b) compares the string  
c) undefined function  
d) copies the string

Answer: b  
Explanation: The strcmp() function compares the string s1 to the string s2.  
int strcmp(const char \*s1,const char \*s2);

**86**. What is the prototype of strcoll() function?  
a) int strcoll(const char \*s1,const char \*s2)  
b) int strcoll(const char \*s1)  
c) int strcoll(const \*s1,const \*s2)  
d) int strcoll(const \*s1)

Answer: a  
Explanation: The prototype of strcoll() function is int strcoll(const char \*s1,const char \*s2).

**87**. What is the function of strcoll()?  
a) compares the string, result is dependent on the LC\_COLLATE  
b) copies the string, result is dependent on the LC\_COLLATE  
c) compares the string, result is not dependent on the LC\_COLLATE  
d) copies the string, result is not dependent on the LC\_COLLATE

Answer: a  
Explanation: The strcoll() function compares the string s1 to the string s2, both interpreted as appropriate to the LC\_COLLATE category of the current locale.

**88**. Which of the following is the variable type defined in header string. h?  
a) sizet  
b) size  
c) size\_t  
d) size-t

Answer: c  
Explanation: This is the unsigned integral type and is the result of the sizeof keyword.

**89**. NULL is the macro defined in the header string. h.  
a) true  
b) false

Answer: a  
Explanation: NULL macro is the value of a null pointer constant.

**90.**What will be the output of the following C code?

const char pla[] = "string1";

const char src[] = "string2";

printf("Before memmove place= %s, src = %s**\n**", pla, src);

memmove(pla, src, 7);

printf("After memmove place = %s, src = %s**\n**", pla, src);

a) Before memmove place= string1, src = string2 After memmove place = string2, src = string2  
b) Before memmove place = string2, src = string2 After memmove place= string1, src = string2  
c) Before memmove place = string2, src = string1 After memmove place= string2, src =string2  
d) Before memmove place= string1, src = string2 After memmove place=string1, src = string1

Answer: a  
Explanation: In the C library function void \*memmove(void \*str1, const void \*str2, size\_t n) copies n characters from str2 to str1.

**91**. What will be the output of the following C code?

const char str1[]="ABCDEF1234567";

const char str2[] = "269";

len = strcspn(str1, str2);

printf("First matching character is at %d**\n**", len + 1);

a) First matching character is at 8  
b) First matching character is at 7  
c) First matching character is at 9  
d) First matching character is at 12

Answer: a

**92**.enum colors {BLACK,BLUE,GREEN}

main()

{

printf( "%d..%d..%d", BLACK, BLUE, GREEN );

return(1);

}

a.1..2..3

b.0..1..2

c.1..1..1

d.0..0..0

**Answer:** B) 0..1..2  
  
**Explanation:**  
enum assigns numbers starting from 0, if not explicitly defined.