TCS NINJA - PROGRAMMING LOGIC SET 2

1. What will be the output? #include <stdio.h>
   int main()
   {
   int main = 5; printf("%d", main); return 0;
   }

a) compile-time error
b) run-time error
c) run without any error and prints 5
d) experience infinite looping
Explanation: The code has no error, it will be compiled easily and produce the output i .e; 5

2. What is the output of this 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
   Explanation: The output will be -128 as the code has signed data type

3. What is the output of this C code? #include <stdio.h>
   int main()
   {
   float x = 'a'; printf("%f", x); return 0;
   }
   a) 97.000000 b) run time error c) a.0000000 d) a
   Explanation: The code will return the ascii value of 'a' in decimal because of float

4. What will happen if the below program 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

Explanation: The code doesn't have any errors, it will compile successfully and print the output 3

5. Which of the following is not valid variable name declaration?

   a) int__ v1; b) int__ 1v; c) int__ V1; **d) None**

   Explanation:

   All **variable names** must begin with a letter of the alphabet or an. underscore(_). ...

   After the first initial letter, **variable names** can also contain letters and numbers. ...

   Uppercase characters are distinct from lowercase characters. ...

   You cannot use a C++ keyword (reserved word) as a **variable name**.

6. Which is not a valid C variable name?

   a) Int number; b) float rate; c) int variable_count; **d) int $main;**

   Explanation: Variable name cannot start with a $ symbol

7. Consider the following function function calculate(n)

   {

   if(n equals 5)

   return 5 else

   return (n + calculate(n-5)) end

   }

   Shishir calls the function by the statement, calculate(20). What value will the function return?

   a) 500 b) 24 c) 20 **d) 50**

   Explanation: its recursive fn.

   calc(20) returns (20+calc(20-5)) = 20+calc(15) = 20+15+calc(10) = 20+15+10+calc(5) = 20+15+10+5 = 50

8. Choose the correct answer: Tanuj writes the code for a function that takes as input n and calculates the sum of first n natural numbers.

   Function sum(n)

   {

   if(??) return 1 else

   return (n + sum(n-1)) end

   }

   Fill in ?? in the code.

   a) n equals 2 **b) n equals 1** c) n >=1 d) n > 1

   Explanation: 3+sum(2) = 3+2+sum(1)

9. What will be the output of the program? #include<stdio.h>
int main()
{
int a[5] = {5, 1, 15, 20, 25};
int i, j, m;
i = ++a[1];
j = a[1]++;
m = a[i++];
printf("%d, %d, %d", i, j, m); return 0;
}
a) 2, 1, 15 b) 1, 2, 5 **c) 3, 2, 15** d) 2, 3, 20
Explanation: Step 1: int a[5] = {5, 1, 15, 20, 25}; The variable arr is declared as an integer array with a size of 5 and it is initiapzed to a[0] = 5, a[1] = 1, a[2] = 15, a[3] = 20, a[4] = 25 .
Step 2: int i, j, m; The variable i,j,m are declared as an integer type.
Step 3: i = ++a[1]; becomes i = ++1; Hence i = 2 and a[1] = 2
Step 4: j = a[1]++; becomes j = 2++; Hence j = 2 and a[1] = 3.
Step 5: m = a[i++]; becomes m = a[2]; Hence m = 15 and i is incremented by 1(i++ means 2++ so i=3)
Step 6: printf("%d, %d, %d", i, j, m); It prints the value of the variables i, j, m
Hence the output of the program is 3, 2, 15

10. What is the output of this 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!
Explanation: %d has no relevance here

11. Consider the following code: function modify(a,b)
{
integer c, d = 2 c = a*d + b return c
}
function calculate( )
{
integer a = 5, b = 20, c integer d = 10
c = modify(a, b); c = c + d
print c
}
a) 80 **b) 40** c) 32 d) 72

Explanation: c=? d=2 c=a*d+b
c=a*2+b
then a=5 b=20 , c=? c=c+d
c=a*2+b+d c=5*2+20+10=40

12. What is the data type that occupies the least storage in "C" language?
    a) int **b) char** c) float d) dunle
    Explanation: **C** is associated with different **data types** to each **variable** and **occupies** different amount of **memory**. The char **data type** is used to **store** a single character and is the most basic **data type** in **C**. It requires only one byte of **memory** for **storage** and can contain both positive and negative values.

13. What is the base case for the following code? void my_recursive_function(int n)
    {
    if(n == 0) return; printf("%d ",n);
    my_recursive_function(n-1);
    }
    int main()
    {
    my_recursive_function(10); return 0;
    }
    a) Return b) printf("%d ", n) **c) if(n == 0)** d) my_recursive_function(n-1)
    Explanation: For the base case, the recursive function is not called. So, "if(n == 0)" is the base case.

14. How many times is the recursive function called, when the following code is executed?
    void my_recursive_function(int n)
    {
    if(n == 0) return; printf("%d ",n);
    my_recursive_function(n-1);
    }
    int main()
    {
    my_recursive_function(10); return 0;
    }
    a) 9 b) 10 **c) 11** d) 12
    Explanation: The recursive function is called 11 times.

15. What will be the output of the following code? int cnt=0;
    void my_recursive_function(int n)
    {
    if(n == 0) return; cnt++;
    my_recursive_function(n/10);

```
}
int main()
{
my_recursive_function(123456789); printf("%d",cnt);
return 0;
}
```

a) 123456789 b) 10 **c) 9** d) 12

Explanation: The program prints the number of digits in the number 123456789, which is 9.

16. What will be the output of the following code?

```
void my_recursive_function(int n)
{
if(n == 0)
{
printf("False"); return;
}
if(n == 1)
{
printf("True"); return;
}
if(n%2==0) my_recursive_function(n/2); else
{
printf("False"); return;
}
}
int main()
{
my_recursive_function(100); return 0;
}
```

a) True **b) False** c) Compile time error d) Run time error

Explanation: The function checks if a number is a power of 2. Since 100 is not a power of 2, it prints false.

17. What is the output of the following code?

```
int cnt = 0;
void my_recursive_function(char *s, int i)
{
if(s[i] == '\0') return;
if(s[i] == 'a' || s[i] == 'e' || s[i] == 'i' || s[i] == 'o' || s[i] == 'u') cnt++;
my_recursive_function(s,i+1);
}
int main()
{
my_recursive_function("thisisrecursion",0); printf("%d",cnt);
```

```
return 0;
}
```

a) 6 b) 4 c) 9 d) 10

Explanation: The function counts the number of vowels in a string. In this case the number is vowels is 6

18. What is the output of the following code?

```
void my_recursive_function(int *arr, int val, int idx, int len)
{
if(idx == len)
{
printf("-1"); return ;
}
if(arr[idx] == val)
{
printf("%d",idx); return;
}
my_recursive_function(arr,val,idx+1,len);
}
int main()
{
int array[10] = {7, 6, 4, 3, 2, 1, 9, 5, 0, 8};
int value = 2; int len = 10;
my_recursive_function(array, value, 0, len); return 0;
}
```

a) 3 **b) 4** c) 5 d) 6

Explanation: The program searches for a value in the given array and prints the index at which the value is found. In this case, the program searches for value = 2. Since, the index of 2 is 4(0 based indexing), the program prints 4.

19. function main() { integer a=5,b=7 switch(a) { case 5 :print "I am 5" break case b:print "I am not 5" break default:print "I am different" } }

a) I am 5 b) I am not 5 c) I am different **d) Error**

Explanation: Break should have; for not showing error

20. Function main()

```
{
Integer i=0.7 Static float m=0.7 If(m equals i)
Print(“We are equal”) Else If(m>i)
Print(“I am greater”) Else
Print(“I am lesser”)
```

a) We are equal b) I am greater c) I am lesser **d) This code will generate an error**

Explanation: two different data types cannot be compared with each other without type-casting

21. puts function adds newline character

a) True
b) Flase
c) Depends on the standard
d) Undefined behaviour

22. puts function adds newline character

a) FILE is a keyword in C for representing files and fp is a variable FILES type
b) FILE is a structure and fp is a pointer to the structure of FILE type
c) FILE is a stream
d) FILE is a buffered stream

Explanation: fp is a pointer of FILE type and FILE is a structure that store following information about opened file.

23. The first and second arguments of fopen are _____

a) A character string containing the name of the file & the second argument is the mode
b) A character string containing the name of the user & the second argument is the mode
c) A character string containing file pointer & the second argument is the mode
d) None of the mentioned

Explanation: The first argument of fopen contains the file name and the second argument contains the mode of the file that whether the file is open, closed, holed etc.

24. fseek() should be preferred over rewind() mainly because

a) rewind() doesn't work for empty files
b) rewind() may fail for large files
c) In rewind, there is no way to check if the operations completed successfully
d) All of the above

Explanation: They both do the same thing in two different ways: they set the pointer to the beginning **of** the file. The only **difference is** that **rewind** also clears the error indicator.

So **fseek()** should be preferred over **rewind()** . This **is** because **rewind** doesn't return an integer indicating whether the operation has succeeded.

25. FILE is of type _____

a) int type b) char * type **c) struct type** d) None of the mentioned

Explanation: It is a data-type containing collected values of various data-types hence it is user generated and of the type struct

26. FILE reserved word is

a) A structure tag declared in stdio.h
b) One of the basic datatypes in c
c) Pointer to the structure defined in stdio.h
d) It is a type name defined in stdio.h

Explanation: Often found in programming languages and macros, **reserved words** are terms or phrases appropriated for special use that may not be utilized in the creation of variable names. For example, "print" is a **reserved word** because it is a function in many languages to show text on the screen.

27. getc() returns EOF when

a) End of files is reached
b) When getc() fails to read a character
c) Both of the above
d) None of the above

Explanation: **getc** is equivalent to fgetc. **getc returns** the next character from the stream referred to by fp; it **returns EOF** for End Of File or error.

28. What will be output of the following program int main()

```
{
int arr[4]={3,4,5,6}; int k[4];
k=arr; printf("%d\n",k[1]);
}
```

a) Compile time error b) 4 c) No output d) Program crashes

Explanation: we cannot copy an array using assignment operator because the c language do not provide an array type instead, array are simply an area of contiguous allocated memory and the name of the array is actually a pointer to the first element of the array.

29. Predict the output of following code: main()

```
{
int x,a=10; x=a==10?printf("hait"):printf("hellon"); printf("%d",x);
}
```

a) hait 4 b) Error c) hello 3 d) hai hello

Explanation: hait,4 as a=10,then x=a=10...so 10==10...which is true..so it will print the if part and and finally print the the number of words in the printf statement.

30. What will happen if in a C program you assign a value to an array element whose subscript exceeds the size of array?

a) **The element will be set to 0.**
b) The compiler would report an error.
c) The program may crash if some important data gets overwritten.
d) The array size would appropriately grow.

Explanation: The element would be set to zero option 1 #include

```
int main()
{
int a[5]={1,2,3,4};int i;
```

printf("%d",a[6]); return 0;
}
output : 0

31. What will be output of the following program int main()
{
int b[4]={5,1,32,4};
int k,l,m;
k=++b[1];
l=b[1]++;
m=b[k++];
printf("%d, %d, %d",k,l,m); return 0;
}
a) 2,3,22 b) 3,2,4 **c) 3,2,32** d) 2,2,4
Explanation: Here, ++b[1] means that firstly b[1] will be incremented so, b[1]=2 then assigned to k i.e. k=2. b[1]++ means firstly b[1] will be assigned to variable l i.e. l=2, Then value stored in b[1] will be incremented i.e. b[1]=3. b[k++] means first b[k] will be assigned to m i.e. m=32, then value of k will be incremented i.e. k=3.

32. What will be output of the following program where c=65474 and int=2 bytes. int main()
{
int c[3][4]={2,3,1,6,4,1,6,2,2,7,1,10};
printf("%u, %u\n", c+1, &c+1); return 0;
}
a) 65482, 65498 b) 65476, 65476 c) 65476, 65498 d) No output
Explanation: Here c[3][4]= {
{2,3,1,6};
{4,1,6,2};
{2,7,1,10}
};
c+1 means c is base address i.e. address of 1st one Dimensional array and on incrementing it by 1 means it points to 2nd one 2 Dimensional array.
So, c+1=65474 + (4 * 2)= 65482
But, when we are writing &c, that means address of this whole array i.e. address of next new array.
So, &c+1=65474 + (12 * 2)=65498

33. What will be output of the following program int main()
{
int a[5],i=0; while(i<5) a[i]=++i; for(i=0;i<5;i++) printf("%d,",a[i]);
}
a) garbage value,1,2,3,4 b) 1,2,3,4,5 c) Error d) program crash

Explanation: firstly right side of any expression is evaluated, then the left side is evaluated. So, here ++i will be evaluated at first, then a[i]. Hence, when i=0, a[1]=1, then i=1, a[2]=2,…a[4]=4 and a[0]=garbage value

34. What will be output of the following program int main()
{
float a[]={12.4, 2.3, 4.5, 6.7};
printf("%d, %d", sizeof(a), sizeof(a[0])); return 0;
}
a) 16 bytes, 4 bytes b) 16 bytes, 3 bytes c) 8 bytes, 4 bytes d) None of them
Explanation: sizeof(a) = number of element * size of each element = 4 * 4 bytes = 16 bytes
sizeof(a[0]) = size of 1st element of array a = 4 bytes

35. The output of this C code is?
#include <stdio.h>
void main()
{
int i = 0; if (i == 0)
{
printf("Hello"); break;
}
}
a) hello is printed infinite times b) hello c) varies **d) compile time error**

36. What is the output of this C code? #include
int *i;
int main()
{
if (i == NULL)
printf("true\n"); return 0;
}
a) true b) true only if null value is 0 c) compile time error d) nothing
Explanation: it will print true because the variable i does not have any value so it will not point any address.

37. What will be the output of the following C code? #include<stdio.h>
void main()
{
int k=4; float k=4;
printf("%d",&k);
}
a) Compile time error b) 4 c) 4.000000000 d) 4 4
Explanation: Since the variable k is defined both as integer and as float, it results in an error. Output:

$ cc pgm8.c
pgm8.c: In function ‘main’:
pgm8.c:5: error: conflicting types for ‘k’
pgm8.c:4: note: previous definition of ‘k’ was here
pgm8.c:6: warning: format ‘%d’ expects type ‘int’, but argument 2 has type ‘double’ pgm8.c:7: error: expected ‘;’ before ‘}’ token

38. Which of the following is not correct for static variables?

a) The value of a static variable in a function is retained between repeated function calls to the same function.
b) static variables are allocated on the heap
c) static variables do not retain its old value between function calls
d) The scope of static variable is local to the block in which it is defined.

Explanation: static variable preserve its value in between function calls.
For example,

Without static:

```
#include<stdio.h> int fun()
{
int count = 0; count++; return count;
}

int main()
{
printf("%d ", fun());
printf("%d ", fun()); return 0;
}
```

OUTPUT:

1 1

With static:

```
#include<stdio.h> int fun()
{
static int count = 0; count++;
return count;
}
int main()
{
printf("%d ", fun());
printf("%d ", fun()); return 0;
}
```

OUTPUT:

1 2

In second case, the variable count preserved its value in between the two function calls(i.e, from main the function fun() called two times).

39. #include <stdio.h> int main()
{
static int i=5; if(--i){
main();
printf("%d ",i);
}
}
a) 4 3 2 1 b) 1 2 3 4 c) 0 0 0 0 d) Compiler error
Explanation: A static variable is shared among all calls of a function. All calls to main() in the given program share the same i. i becomes 0 before the printf() statement in all calls to main().

40. Output of following program?
#include <stdio.h>
int main()
{
static int i=5; if (--i){
printf("%d ",i); main();
}
}
a) 4 3 2 1 b) 1 2 3 4 c) 0 0 0 0 d) Compile time error
Explanation: Since i is static variable, it is shared among all calls to main(). So is reduced by 1 by every function call.

41. Output of following program?
#include <stdio.h>
int main()
{
int x = 5;
int * const ptr = &x;
++(*ptr); printf("%d", x); return 0;
}
a) Compile time error b) Run time error **c) 6** d) 5
Explanation: See following declarations to know the difference between constant pointer and a pointer to a constant. **int * const ptr** —> ptr is constant pointer. You can change the value at the location pointed by pointer p, but you can not change p to point to other location. **int const * ptr** —> ptr is a pointer to a constant. You can change ptr to point other variable. But you cannot change the value pointed by ptr. Therefore above program works well because we have a constant pointer and we are not changing ptr to point to any other location. We are only icrementing value pointed by ptr.

42. What is the output? #include<stdio.h> int main()
{
typedef static int *i; int j;
i a = &j; printf("%d", *a); return 0;
}
a) Run time error b) Garbage value **c) Compiler error** d) Time error
Explanation: Compiler Error -> Multiple Storage classes for a. In C, typedef is considered as a storage class. The Error message may be different on different compilers.

43. Consider the following C function
int f(int n)
{
static int i = 1; if (n >= 5)
return n; n = n+i; i++;
return f(n);
}
a) 5 b) 6 **c) 7** d) 8
Since i is static, first line of f() is executed only once. Execution of f(1)
i = 1
n = 2
i = 2 Call f(2) i = 2
n = 4
i = 3 Call f(4) i = 3
n = 7
i = 4 Call f(7)
since n >= 5 return n(7)

44. #include <stdio.h> int main()
{
int i = 1024; for (; i; i >>= 1)
printf("GeeksQuiz"); return 0;
}
How many times will GeeksQuiz be printed in the above program?
a) 10 **b) 11** c) Infinite d) None
Explanation: In for loop, mentioning expression is optional. **>>=** is a composite operator. It shifts the binary representation of the value by 1 to the right and assigns the resulting value to the same variable. The for loop is executed until value of variable **i** doesn't drop to 0.

45. Predict the output of the above program? #include<stdio.h>
int main()
{

```
int n;
for (n = 9; n!=0; n--)
  printf("n = %d", n--); return 0;
}
```

a) 9 7 5 3 1 b) 9 8 7 6 5 4 3 2 1 **c) Infinite loop** d) 9 7 5 3

Explanation: The program goes in an infinite loop because n is never zero when loop condition (n != 0) is checked. n changes like 7 5 3 1 -1 -3 -5 -7 -9 ..

46. Predict the output of the below program:

```
#include <stdio.h>
#define EVEN 0
#define ODD 1 int main()
{
int i = 3; switch (i & 1)
{
case EVEN: printf("Even");
break;
case ODD: printf("Odd"); break;
default: printf("Default");
}
return 0;
}
```

a) Even **b) Odd** c) Default d) Compile time error

Explanation: The expression i & 1 returns 1 if the rightmost bit is set and returns 0 if the rightmost bit is not set. As all odd integers have their rightmost bit set, the control goes to the block labelled ODD.

47.

```
#include <stdio.h>
int i;
int main()
{
if (i); else
   printf("Ëlse"); return 0;
}
```

What is the output of the above program?

a) if block is executed
b) else block is executed
c) It is unpredictable as i is not initialized
d) Error: misplaced else

Explanation: Since i is defined globally, it is initialized with default value 0. The Else block is executed as the expression within if evaluates to FALSE. Please note that the empty block is equivalent to a semi-colon(;). So the statements if (i); and if (i) {} are equivalent.

48. The output of the code below is #include <stdio.h>
void main()
{
int a = 5; if (true);
printf(“hello”);
}

a) It will display hello
b) It will throw an error
c) No output
d) Depends on compiler

49. The output of the code below is #include <stdio.h>
void main()
{
int a = 0; if (a == 0) printf(“hi”); else
printf(“how are u”); printf(“hello”);
}

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

50. What will be output of the following program if argument passed to command lines are : prog 1 4 2
#include<stdio.h>
int main(int argc, char *argv[])
{
while(argc–) printf(“%s\n”,argv[argc]); return 0;
}
a) 2, 4, 1
b) Garbage-value 2 4 1
c) Garbage-value 2 4 1 prog
d) Infinite Loop

51. What will be the output of the program (sample.c) given below if it is executed from the command line (Turbo C in DOS)?
cmd> *sample 1 2 3*
>#include<stdio.h>
int main(int argc, char *argv[])
{
int j;
j = argv[1] + argv[2] + argv[3]; printf("%d", j);

return 0;
}

a) 6
b) Sample 6
c) error
d) garbage value

Explanation: here argv[1], argv[2] and argv[3] are string type. We have to convert the string to integer type before perform arithmetic operation.

52. What will be the output of this program? #include<stdio.h>
int main(int argc, cahr *argv[])
{
printf("%d",argc); return 0;
}

a) 4
b) **5**
c) 6
d) 3

Explanation: The first argument of main **argc** contains the total number of arguments given by command prompt including command name, in this command total arguments are 5.

53. Which is the correct form to declare main with command line arguments ?

a) int main(int argc, char argv[]){ }
b) int main(int argc, char* argv[]){ }
c) int main(int argc, char **argv){ }
d) int main(int* argc, char* argv[]){ }

Explanation: int main(int argc, char* argv[]){ }

54. Which of the following syntax is correct for command-line arguments?

a) int main(int var, char *argv[])
b) int main(char *arv[], int arg)
c) int main(char c,int v)
d) int main(int v,char c)

55. What will be the output of following program ? #include <stdio.h>
void main()
{
printf("includehelp.com\rOK\n"); printf("includehelp.com\b\b\bOk\n");
}

a) OK includehelp.ok
b) OK includehelp.okm
c) includehelp.com includehelp.okm
d) OKcludehelp.com includehelp.okm

Explanation: OKcludehelp.com includehelp.Okm
/r is an escape sequence which means carriage return. Carriage return takes back the cursor to the leftmost side in a line.
Thus in the statement printf("includehelp.com\rOK\n");
First **"includehelp"** is printed (not still displayed) then cursor moves to leftmost position (**"i"** here) and starts printing **"OK"** which results in overwriting of first two characters
of **"includehelp"**. Thus the final output is **"OKcludehelp.com"** and then cursor moves to next line due to character feed \n.
In the second statement /b escape character is used which is equivalent to backspacing the cursor. Overwrite also took place here due to three backspaces.

56. What will be the output of following program ?

```
#include <stdio.h>
void main(){ unsigned char c=290; printf("%d",c);
}
```

a) 34
b) 290
c) Garbage
d) Error

Explanation: 290 is beyond the range of unsigned char. Its corresponding value printed is: (290 % (UINT_MAX +1) where UINT_MAX represents highest (maximum) value of UNIT type of variable. Here it's character type and thus UINT_MAX=255 Thus it prints 290 % (UINT_MAX +1)=34

57. What will be the output of following program ? #include <stdio.h>

```
void main(){ int a=0; a=5||2|1; printf("%d",a);
}
```

a) **1**
b) 7
c) 8
d) 9

Explanation: 5 || 2 | 1 is actually 5 || (2 |1) Now
2= 0000 0010
1= 0000 0001
2|1= 0000 0011=3
5 || 3 returns true as both are nonzero Thus a=1

58. What will be the output of following program (on 32 bit compiler)? #include <stdio.h>

```
int main(){ float a=125.50; int b=125.50; char  c='A';
printf("%d,%d,%d\n",sizeof(a),sizeof(b),sizeof(125.50));
printf("%d,%d\n",sizeof(c),sizeof(65));
return 0;
}
```

a) 4,4,4 1,4
b) 4,4,8 1,4
c) 4,4,4 1,1
d) 4,4,8 1,4

Explanation:
sizeof(a)= 4 bytes (float), sizeof(b)=4 bytes(int...in Visual studion, 2 in turboc), sizeof(125.50)=8 bytes (125.50 will consider as double constant), sizeof(c)= 1 byte (character), sizeof(65) = 4 bytes (65 is an integer).

59. What will be the output of following program ? #include <stdio.h>

```
int main(){ static char a; static long b; int c;
printf("%d,%d,%d",a,b,c);
return 0;
}
```

a) 0,0,0
b) Garbage,Garbage,Garbage
c) Garbage,Garbage,0
d) 0,0,Garbage

Explanation: Static variable always initialize with 0, and other one by garbage value.
N.B: output is compiler dependent

60. What will be the output of following program?

```
#include <stdio.h>
int main()
{
int ok=-100;
-100;
printf("%d",ok);
return 0;
}
```

a) 100
b) error
c) **-100**
d) 101

Explanation: -100 is evaluated and this does not effect the value of **ok**

61. What will be the output of following program ? #include <stdio.h>

```
enum numbers
{
zero, one, two, three , four=3,five,six,seven=0,eight
};
void main()
{
printf("%d,%d,%d,%d,%d,%d,%d,%d,%d",zero,one,two,three,four,five,six,seven,eight);
}
```

a) **0,1,2,3,3,4,5,0,1**
b) 0,1,2,4,5,6,7,8,9
c) 0,1,2,3,3,1,2,3,4
d) 0,1,2,3,3,4,5,0,9

Explanation: -0,1,2,3,3,4,5,0,1
We have discussed that enum values starts with 0 (if we do not provide any value) and increase one by one. We are assigning 3 to four, then four will be 3, five will be 4 ... again we are assigning 0 to seven then seven will be 0 and eight will be 1.

62. What is the output of this C code? #include <stdio.h>

```
void main()
{
int k = 5; int *p = &k;
int **m = &p; printf("%d%d%d\n", k, *p, **p);
}
```

a) 5 5 5
b) 5 5 junk
c) 5 junk junk
d) Compile time error

63. Which of the following statements about stdout and stderr are true?
a) They both are the same
b) Run time errors are automatically displayed in order
c) Both are connected to the screen by default
d) stdout is line buffered but stderr is unbuffered

64. Given the below statements about C programming language;
a) main() function should always be the first function present in a C program
b) all the elements of an union share their memory location
c) A void pointer can hold address of any type and can be type casted to any
d) A static variable hold random junk value if it is not initialized

65. If a function is defined as static, it means
a) The value returned by the function does not change
b) all the variable declared inside the function automatically will be assigned
c) It should be called only within the same source code/program file
d) None of the other choices as it is wrong to add static prefix to a function

66. Comment on the below while statement while (0 == 0) { }
a) It has syntax error as there are no statements within braces { }
b) It will run forever
c) It compares 0 with 0 and since they are equal it will exit the loop
d) It has syntax error as the same number is being compared with itself

67. Given the below statements about C programming language
a) main() function should always be the first function present in a C program file
b) all the elements of an union share their memory location
c) A void pointer can hold address of any type and can be type casted to any type
d) A static variable hold random junk value if it is not initialized

68. Which of the above are correct statements?
a) 2, 3 b) 1, 2 **c) 1, 2, 3** d) 1, 2, 3, 4

69. What makes a class abstract?
a) By making all members functions constant
b) By making at least one member function as pure virtual function
c) By declaring it abstract using the static keyword
d) By declaring it abstract using the virtual keyword

70. Which type of class allows only one object of it to be created?
a) Virtual class b) Abstract class **c) Singleton class** d) Friend class

71. Which of the following concepts of OOPS means exposing only necessary information to client?
a) Multi-level inheritance b) Multiple inheritances
c) Hybrid inheritance **d) All of the above**

Explanation: Data hiding is a method used in object-oriented programming to hide information within computer code. Objects inside code are not privy to information which is considered hidden. It presents many advantages for programmers because objects are unable to connect to irrelevant data and hackers are less likely to access data.

Data hiding takes specific parts of the code and then conceals those parts from the objects. The objects are not directly accessible in any data that is hidden. If an object is able to access hidden data, it will return an error, and this is because the object cannot see the data.

72. Which type of inheritance needs a virtual function:

a) Friend function **b) Static function** c) Destructor d) None of the above

73. Which of the following cannot be inherited?

a) Public data members b) Private data members

c) Protected data members d) Members functions

74. Which of the following are available only in the class hierarchy chain?

a) Polymorphism b) Structure **c) Inheritance** d) Cascading

75. User perform following type of operation on stack of Size 5:-

push(10) pop() push(5) push(6) push(8) pop() push(9) pop()
pop() push(8)

So how many element is left in stack

a) 1 b) 2 c) 3 d) 4

Explanation:

Stack: [10] (push)
Stack: [ ] (pop)
Stack: [5] (push)
Stack: [5,6] (push)
Stack: [5,6,8] (push)
Stack: [5,6] (pop)
Stack: [5,6, 9] (push)
Stack: [5,6] (pop)
Stack: [5] (pop)
Stack: [5.8] (push)

76. User perform following type of operation on stack of Size 5:-

push(5) push(7) pop() push(9) pop()
pop() push(9) pop() push(1) push(5) pop()
pop() push(4) pop()
pop() push(6)

So which of the following statement is true for the stack

a) Underflow occurs b) Overflow occurs c) The process goes smoothly d) None of them

Explanation: When we push and pop the element so while doing it one condition arises where we are not able to pop any element that arises underflow condition.

77. In which order the elements are removed from the stack.

a) Reverse b) Alternative c) Sequential d) Hierarchical

Explanation: In which way we push an element on the stack as same ways we delete the element but opposite in direction because it works on LIFO.

78. In which place the dequeue operation is performed.
a) Rare b) Top **c) Front** d) Last
Explanation: In queue element are added from rare and delete from the front because it follows first in first out.

79. Which of the following is an application of the stack?
a) Infix to postfix b) Tower of Hanoi c) Finding factorial **d) All of above**
Explanation: These all are the application of the stack

80. What type of data structure is used in the consumer-producer problem? when resource shared with multiple users.
a) Stack b) Linked List **c) Queue** d) None of the above
Explanation: The queue is used to share resources among multiple users.

81. Which of the following code is used to create a new node.
a) ptr = (Node*)malloc(sizeOf(node));
b) ptr=(Node*)malloc(node);
c) ptr=(Node*)malloc(soxeOf(node*));
d) ptr=(Node)malloc(sizeOf(node));
Explanation: this is the syntax for the memory allocation.

82. What type of memory referred for the linked list.
a) Static memory allocation
b) Local memory
c) Dynamic memory allocation
d) All types of memory can assigned
Explanation: In a link list, memory assign at run time.

83. The linked list used to implement
a) Stack
b) Queue
c) Graph
d) All of these
Explanation: With the help of link list we can implement these all data structure.

84. What are the application that is use the linked list?
a) Queue
b) Stack
c) Both a and b
d) None of them
Explanation: stack and queue both are the application of the linked list.

85. What is the biggest advantage of the linked list?

a) There is no need to specify fixed size of linked list

b) Add only certain element

c) Add only specify data

d) None of them

Explanation: In linked list the size is automatically increase when we perform insertion and size is automatically reduced when the element is deleted.

86. In linked list implementation, what type of field a node carries?

a) Data

b) Link

c) Both a and b

d) None of them

Explanation: In linked list there are two field, first field holds data and second field holds link of another node.

87. What is the recurrence relation formed for the complexity of binary search?

a) T(n) = T(n/2) + log n

b) T(n) = T(n/2) + k, k is constant

c) T(n) = T(n/2) + n

d) T(n) = 2T(n/2) + k, k is constant

Explanation: The recurrence relation for complexity is T(n)=T(n/2) + k

88. The children of the same parent are known as

a) Leaf node

b) Non-leaf node

c) Adjacent node

d) Siblings

Explanation: The parent who have two children, these children known as the siblings.

89. Which of the following is not a height balanced tree?

a) Binary search tree

b) Binary heap tree

c) splay tree

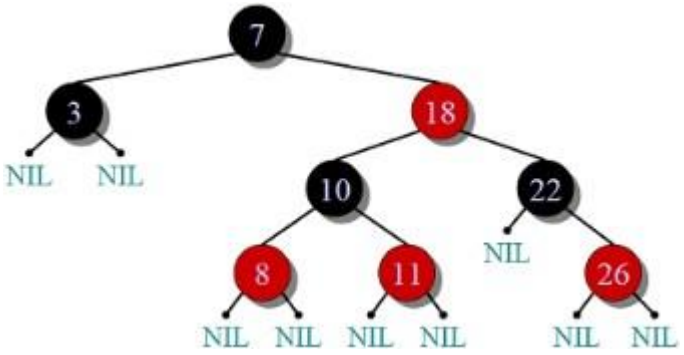
d) AVL tree

Explanation: In the binary search tree, we just put the value either left or right side. So it is not a height-balanced tree.

90. In which tree a leaf node carries null value

a) Complete tree **b) Red black tree** c) Splay tree d) Binary search tree

Explanation: This is a red-black tree, here leaf node has a null value



91. In which traversal root node is visit at the starting?

a) Preorder

b) Postorder

c) Inorder

d) None of these

Explanation: In preorder, we follow this sequence. Root Left Right. So in the preorder first root node traverses.

92. The expression which is used in a programming language.?

a) Postfix

b) Prefix

c) Infix

d) Depend on Language

Explanation: In programming language, infix order is used.

93. In which searching algorithm Backtracking allowed?

a) Binary Search

b) DFS

c) BFS

d) All of the above is true

Explanation: Backtracking can be defined as a general algorithmic technique that considers searching every possible combination in order to solve a computational problem. So DFS using backtracking to find shortest path to search node.

94. In a graph, what does the mean from **E=(u,v).?**

a) u is adjacent to v but v is not adjacent to u

b) e begins from u and ends with v

c) u is processor and v is successor

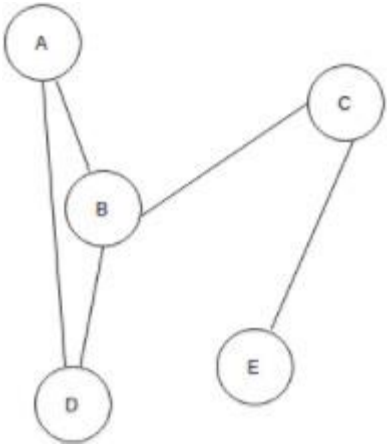
d) both B and C

Explanation: E=(u,v) There is E for edges u is starting node

v is end node.

Mean the edge E connected with node u to v with directed edge

95. In the following graph identify cut vertices.



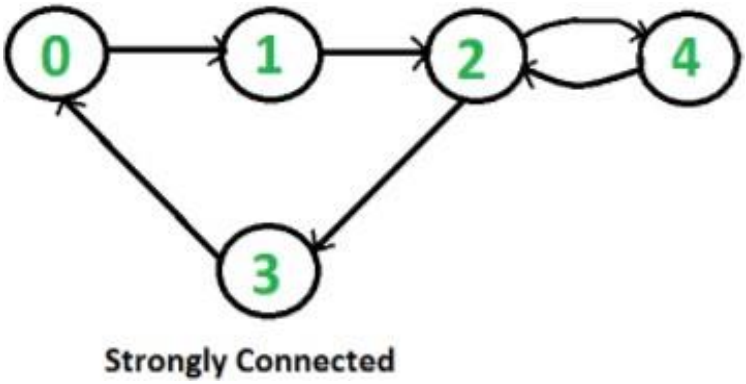
a) B and E
b) C and D
c) B and C
d) A and B

Explanation: Here if we cut the edge between B and C so it make two separate graph

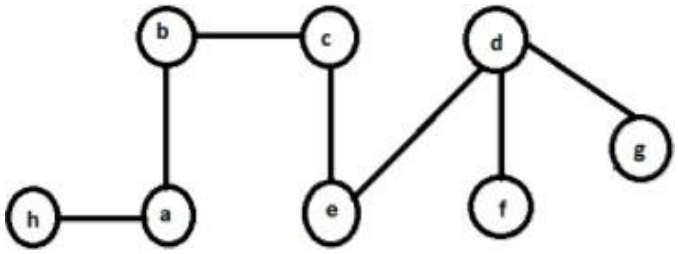
96. How is the directed graph connect, if there is a path from each vertex to every other vertex in the digraph?

a) Tightly connected
b) Loosely connected
c) Strongly connected
d) Weakly connected

Explanation: The graph is strongly connected if there is a path between any two pairs of vertices. For example, the following graph is strongly connected

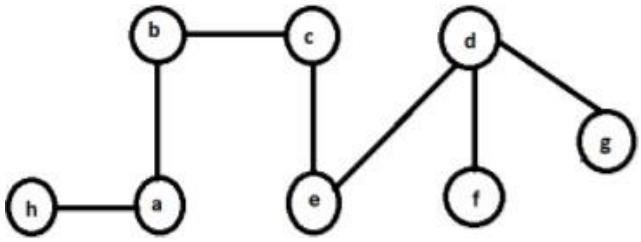


97. What will be the order in which the vertices will be traversed, if BFS is applied on the graph, starting from the vertices b?



a) bachedgf
b) bcaedfgh
c) both A and B
d) Neither A nor B
Explanation: Queue= { b } or { b } Queue={ b,a,c } or { b,c,a }
Queue= { b,a, c,h } or { b,c,a,e }
Queue= { b,a,c,h, e } or { b,c,a,e }
Queue= { b,a,c,h, e,d } or { b,c,a,e,d }
Queue= { b,a,c,h, e,d,g,f } or {b,c,a,e,d,f,g, h,} Sequence= (bachedgf) or (bcaedfgh)

98. What is the minimum time is required to visit the node f by using DFS? the starting node is b.



a) 6
b) **5**
c) 4
d) 7
Explanation: b is starting node
Stack : b/0/15 ,c/1/10, e/2/9, d/3/8, f/4/5 ,g/6/7, a/11/14, h/12/13 So node f take 5 time to visit

99. In simple chaining, what data structure is appropriate?
a) Singly linked list
b) Doubly linked list
c) Circular linked list
d) Binary trees
Explanation: Deletion becomes easier with doubly linked list, hence it is appropriate.

100. In simple uniform hashing, what is the search complexity?
a) O(n)
b) O(logn)

c) O(nlogn)

d) O(1)

Explanation: There are two cases, once when the search is successful and when it is unsuccessful, but in both the cases, the complexity is O(1+alpha) where 1 is to compute the hash function and alpha is the load factor.

101. Given a hash table T with 25 slots that stores 2000 elements, the load factor α for T is _____ .
    a) 8000
    b) 0.0125
    c) 80
    d) 1.25
    Explanation: load factor = (no. of elements) / (no. of table slots) = 2000/25 = 80

102. A hash table of length 10 uses open addressing with hash function h(k)=k mod 10, and linear probing. After inserting 6 values into an empty hash table, the table is as shown below.
    a) 46, 42, 34, 52, 23, 33
    b) 34, 42, 23, 52, 33, 46
    c) 46, 34, 42, 23, 52, 33
    d) 42, 46, 33, 23, 34, 52
    Explanation: The sequence (A) doesn't create the hash table as element 52 appears before 23 in this sequence.
    The sequence (B) doesn't create the hash table as element 33 appears before 46 in this sequence.
    The sequence (C) creates the hash table as 42, 23 and 34 appear before 52 and 33, and 46 appears before 33.
    The sequence (D) doesn't create the hash table as element 33 appears before 23 in this sequence.

103. What is it called when several elements compete for the same bucket in the hash table?
    a) Diffusion
    b) Replication
    c) Collision
    d) None of the mentioned
    Explanation: By definition

104. How many different insertion sequences of the key values using the same hash function and linear probing will result in the hash table shown above?

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

a) 10
b) 20
c) 30
d) 40

Explanation: In a valid insertion sequence, the elements 42, 23 and 34 must appear before 52 and 33, and 46 must appear before 33.
Total number of different sequences = **3! x 5** = 30
In the above expression, **3!** is for elements 42, 23 and 34 as they can appear in any order, and **5** is for element 46 as it can appear at 5 different places.

105. Binary search algorithm uses
a) Linear way to search element
b) Divide and conquer method
c) Bubble sort technique
d) All of them

Explanation: Binary search algorithm only check the middle index of element and if element not found so it divides the array. Now it checks further as same.

106. Which of the following disadvantage of the linear search
a) Require more space
b) Not easy o understand
c) Greater time complexity compared to other searching algorithms
d) All of the mentioned

Explanation: It checks every element in the array so it takes maximum time.

107. It checks every element in the array so it take maximum time.
a) Processor and memory
b) Complexity and capacity
c) Time and space
d) Data and space

Explanation: Time and space is the main key for the efficient algorithms.

108. Which of the following technique is used for finding a value in an array?
a) Bubble sort
b) Binary search algorithm
c) Linear search algorithm
d) All of them

Explanation: These all technique used for the purpose of finding value in the array.

109. Which of the following is not a limitation of the binary search?
a) Must use a sorted array
b) requirement of sorted array is expensive when a lot of insertion and deletion are needed 33.33%
c) there must be a mechanism to access middle element directly16.67%
d) **binary search algorithm is not efficient when the data element more than 150033.33%**
Explanation: binary search can be applied on the large array.

110. The worst case occurred in the linear search algorithm when
a) The element in the middle of an array
b) Item present in the last
c) Item present in the starting
d) Item has maximum value
Explanation: If the element situated at the end of the array, so it takes maximum time to search for that of the element.

111. What is the time, space complexity of following code:

```
int a = 0, b = 0;
for (i = 0; i < N; i++)
{
a = a + rand();
}
for (j = 0; j < M; j++)
{
b = b + rand();
}
```

a) O(N*M) time, O(1) space
b) O(N+M) time, O(N+M) space
c) O(N+M), O(1) space
d) O(N*M) time, O(N+M) space
Explanation: The first loop is O(N) and the second loop is O(M). Since we don't know which is bigger, we say this is O(N + M). This can also be written as O(max(N, M)).
Since there is no additional space being utilized, the space complexity is constant O(1).

112. What is the time complexity of the following code: int a = 0, i = N;

```
while (i > 0) { a += i;
i /= 2;
}
```

a) O(N)
b) O(Sqrt(N))
c) O(N/2)
d) O(log N)

Explanation: We have to find the smallest x such that N / 2^x N x = log(N)

113. What is the time complexity of the following code:

```
int i, j, k = 0;
for (i = n / 2; i <= n; i++)
{
for (j = 2; j <= n; j = j * 2)
{
k = k + n / 2;
}
}
```

a) O(n)
b) O(nLogn)
c) O(n^2)
d) O(n^2Logn)

Explanation: If you notice, j keeps doubling till it is less than or equal to n. A number of times, we can double a number until it is less than n would be log(n).
Let's take the examples here. for n = 16, j = 2, 4, 8, 16
for n = 32, j = 2, 4, 8, 16, 32
So, j would run for O(log n) steps. i runs for n/2 steps.
So, total steps = O(n/ 2 * log (n)) = **O(n*logn)**

114. What is the time complexity of the following code:

```
int a = 0;
for (i = 0; i < N; i++)
{
for (j = N; j > i; j--)
{
a = a + i + j;
}
}
```

a) O(N)
b) O(N*log(N))
c) O(N*log(N))
d) O(N*N)

Explanation: The above code runs total no of times = N + (N – 1) + (N – 2) + … 1 + 0
= N * (N + 1) / 2 = 1/2 * N^2 + 1/2 * N O(N^2) times.

115. The complexity of merge sort algorithm is
    a) O(n)
    b) O(log n)
    c) O(n2)
    d) O(n log n)
    Explanation: The worst case complexity for merge sort is O(n log n)

116. Which of the following case does not exist in complexity theory?
    a) Best case
    b) Worst case
    c) Average case
    d) Null case
    Explanation: The null case does not exist in complexity theory.

117. The worst-case complexity of quicksort is
    a) O(n)
    b) O(log n)
    c) O(n2)
    d) O(n log n)
    Explanation: The worst-case complexity of quicksort is $O(n^2)$.

118. Which is the following searching algorithms works on the divide and conquer?
    a) Binary search
    b) Linear search
    c) Sequential search
    d) All of the above
    Explanation: Binary search only check the middle element of the array. When it check the middle element and middle element is not equal to the given element then we divide list in two sub part.

119. What is the total number of the comparison done in the binary search
    a) n
    b) n^2
    c) n+1
    d) log(N+1)
    Explanation: It takes very less time in there is 6 element so it take log(6+1) that is 0.845098.

120. What is the load factor?
    a) Average array size
    b) Average key size
    c) Average chain length
    d) None of the mentioned

Explanation: In simple chaining, the load factor is the average number of elements stored in a chain and is given by the ratio of a number of elements stored to the number of slots in the array.

121. Predict the output of following code: main()

```
{
int i=10; printf("%d,%d",++i,++i);
}
```

a) 11,12

b) 10,11

c) 10,12

d) 11,15

Explanation: In the above program output should (11,12) because it follows pre-increment