* Overview of C *

1: Given the basic structure of C program.
Asn: Sample Structure of C is given below,

```
main( )
{
printf("Rana");
}
```

Basic structure of C program,

<table>
<tr><td>Documentation section</td></tr>
<tr><td>Link section</td></tr>
<tr><td>Definition section</td></tr>
<tr><td>Global declaration section</td></tr>
<tr><td>main ( )
{

}</td></tr>
<tr><td>Subprogram section

<table><tr><td>Function 1</td></tr><tr><td>Function 2</td></tr><tr><td>----</td></tr><tr><td>Function</td></tr></table>(User-defined function)</td></tr>
</table>

2: State where the following statement are true or false.

a) Every line a C program should end with semicolon. (false)
b) In C lowercase letters are significant. (true)
c) Every C program ends with an END word. (false)
d) main ( ) is where the program begins its execute. (true)
e) A line in a program may have more than one statement. (true)
f) A printf statement can generate only one line of output. (true)
g) The closing brace of the main ( ) in a program is the logical end of the program. (true)
h) Every C program must have at least one user-define function. (true)

i) Only one function may be named main ( ). (true)

3: Fill in the blanks with appropriate words in each of the following statements.

a) Every program statement in a C program must end with a semicolon.
b) The printf function is used to display the output on the screen.
c) The math.h header file contains mathematical function.

* Constants, Variables, and Data Types *

4: What is C token? Give different types of C tokens with example.
Ans: In C every meaningful individual units are called as C tokens. C has six types of tokens that are given below,

C TOKENS

Key words Exam: float while	Constants -10.5 12	identifiers Exam: main amount		String “abc” “year”	Operators + - * /	Special symbols { } [ ]

5: What is key word? Write the ANSI C key words.
Ans: In C all key word’s have fixed meaning and the meaning cant not be change. Key words are called the building blocks for program statements. All key words must be written in lowercase. int, char, long double etc are the example of ANSI C key words.

6: What is identifiers? Given some rules for identifiers.
Ans: Identifiers are the name of variables, functions and arrays that are named by user. The identifiers ‘munim’ and ‘MUNIM’ are completely different from each other. The underscore (-) character may be used in the middle or the starting of an identifier. The (-) used for linking between two in long identifiers. The valid identifiers are given below:
Parg_ram, program, progra_m.

7: What is constant? Given the basic type of C constant.
Ans: The values which are unchangeable are called constants. Basically there are four types of constants in C. They are integer, floating-point constants, character

constants and string constants. C supports different type of constants. They are given below:

Constant			
Numeric constant		Character constant	
Integer constant	Real constant	String character constant	String constant

8: What do you know about integer constant? Explain with example.
Ans: An integer constant means an integer valued number. If constants of a sequence of digits. There are three types of integer constants, decimal, octal and hexadecimal. Some valid types integer are shown below:

Decimal integer : 124, 345;
Octal integer : 0, 0765;
Hexadimal integer: 0X7FF,0XBCD;

9: What is real constants? Give some examples of real constant.
Ans: Real constant a fractional parts or an exponent or both of two. Some examples of real constant are given below:
45.56, 67.89,1.6E-12;

10: What is variable? Write the rules of given a variable name.
Ans: Variable is used to store value in the memory. It passes the information to the function as an argument. In C there are some rules of giving a variable name. They are given below:

Type	keyword
Character data	char
Signed whole numbers	int
Floating-point number	float

11: What is data type? Write three classes of data types which ANSI C supports.
Ans: Data type means different types of data. Each of them is different in size. ANSI C supports three classes of data types. They are primary data type, derived data type, user defined data types.

12: Why do we need declare variables?
Ans: Declaration a variable tells the compiler what type of variable is being used.

13: Explain the declaration of variables with examples.

Ans: Declaration of a variables tells the compiler the name of the variable and what type of data the variable will hold. General form of variable declaration:

Type variable name;
Example:
int x;

Where **int** is a variable type and **x** is variable name.

14: State where the following statements are true or false.

a) Any valid printable ASCII character can be used in an identifier. (true)
b) All variables must be given a type when they are declared. (true)
c) ANSI C treats the variables name and Name. (false)
d) Declaration can appear in any where in C program. (true)
e) The underscore can be used anywhere in an identifier. (true)
f) The keyword void is a data type in C. (true)

* Operators and Expressions *

15: Write the different categories of C operators.
Ans: there are different types of C operators. They are given below:

- Arithmetic operators.
- Relational operators.
- Logical operators.
- Assignment operators.
- Increment and decrement operators.
- Conditional operators.
- Bitwise operators.
- Special operators.

16: Explain relational operators with example.
Ans: Relational operators create relation between two numbers. Explain the relational given below:
a>m is true;
>operator creates a relation between a and m.

17: Explain logical operators with examples.
Ans: Logical operators create logical operations between two expressions. Such as,

If (a>m&&e<0)
Where && operators create a logic between a>m and e<0

18: Explain assignment operators with examples.

Ans: Assignment operators are used to assign the result expression to a variable. Such as,
M=a+2;
Where '=' is assignment operator.

19: Explain increment and decrement operators with examples.

Ans: + + Operators is called increment operator it adds 1 to the operand. Such as:
+ +m or m+ +;
The operators - - is called decrement operator. It subtracts 1 to the operand.
Such as: - -m or m- -;

20: What is the different between + +m and m+ + ?

Ans: When + +m and m+ + are used in expressions on the right-hand side of an assignment statement. Consider the following:

m=4;
a= + +m;

first increase the value of m as 5 then it assigns it's value to the a. Both-hand side is 5.

m=4;
a= m + +;

first the value of m as 4 is assign to a. Then the value of m is increase. The value of left-hand side is 4 and right-hand side is 5.

21: What do you know about conditional operator ?

Ans: The general form of conditional operators is x>y ? 3:5; If x>y is true the result will be 3 else the result will be 5.

22: What do you know about precedence of arithmetic operators ?

Ans: An arithmetic expression will be executed from left to right using the rules of precedence of operators. There are two types of priority levels of arithmetic operators in C,

High priority	* / %
Low priority	+ -

23: What do you know about comma operator ?

Ans: Comma operator is used for relating different expression together. Such as

X=(m=4, n=3, m + n);

24: State where the following statement are true or false.

a) All arithmetic operators have the same level of precedence. (false)
b) The modulus operators % can be used only with integer. (true)
c) The operators <=, <= and != all enjoy the same level of priority. (true)
d) During modulo division, the sign of the result is positive, if both the operands are of the same sign. (true)
e) In C, if a data item is zero, it is consider false. (true)
f) The expression !(x<=y) is same as the expression x>y. (true)
g) An explicit case can be used to change the expression. (true)

25: Find out the output of the following program.

```
Main ( )
{
 int x=100;
 printf("%d\n",10 + x++);
 printf("%d\n",10 +  ++x);
}
```

Output:

110
112

26: What is the output of the following program ?

```
Main ( )
{
 int x=100, y=200;
 printf("%d",(x>y)?x:y);
}
```

Output:

200

27: What do you know about function? Explain the getchar( ) function with an example.

Ans: Functions are the building blocks of C. A function is called by other parts of the program for solving different type of works. C program consist of one or more functions.

The general form of getchar ( ) :

```
char  variable_name;
variable_name = getchar ( );
```

It assigns the value of the variable. getchar ( ) get a single character and then assign to the variable.
For example:
char x;
x= getchar ( );

28: State where the following statement are true or false.

a) The purpose of the header file < stdio.h> is to store the programs created by the user. (false)
b) The C standard function that receives a single character from keyboard is getchar. (true)
c) The getchar cannot be used to read a line of a text from the keyboard. (false)
d) The input list in a scanf statement can contain one or more variable. (true)
e) The scanf function cannot be used to read a single character from the keyboard. (false)
f) The printf statement can contain function calls. (true)

29: State the outputs produced by the following printf statement.

a) Printf(" %d %c %f",10, 'x',1.23);
b) Printf("%2d %c %4.2f",1234, 'x', 1.23);
c) Printf("%d\t %4.2",1234,456);
d) Printf("%d %d %d",10,20);

Output:

a) 10 x1.23
b) 1234 x 1.23
c) 1234 456.00
d) 10 20

30: State errors, if any, in the following statements.

a) Scanf("%c%f%d",city,&price,&year);
b) Scanf("%s%d",city,amount);
c) Scanf("%f%d",&amount,&year);
d) Scanf("\n%f",root);
e) Scanf("%c %d %d",*code,&count,Root);

Ans:

a) Scanf("%c%f%d",&city,&price,&year);
b) Scanf("%s%d",city,&amount);
c) Scanf("%f%d",&amount,&year);
d) Scanf("\n%f",&root);
e) Scanf("%c %d %d",&code,&count,&Root);

**** Decision making and branching ****

31: Write different type of decision making statements.
Ans: there are four types of decision making statements. They are given below:

1. if statement.
2. switch statement.
3. conditional operator statement.
4. goto statement.

32: What do you know about if statement ?

Ans: if statement is used for solving conditional statement. if's works is a logical expression is true or false. The general form of it :

```
if  (test expression)
{
    Statement-block;
 }
Statement-x;
```

33: Write different types of if else statement.

Ans: there are three type of if statement. These are given below:

1. Simple if statement.
2. if-else statement.
3. nested if-else statement.
4. else if statement.

34: Write the general form of simple if statement.

Ans: the general form of simple if statement is given below:

```
if(test expression)
{
statement block;
}
statement x;
```

35: Write the general form of if else statement.

Ans: the general form of if-else statement is given below:

```
if(test expression)
{
statement 1;
statement 2;
----------;
statement n;
```

```
}
else
statement m;
```

36: What do you know about nesting of if else statement ?

Ans: When an if else statement is used into an if else statement then it is called nesting if else statement. Here an example is given below:

```
if(a>b)
{
if(a>c)
printf("a");
else
printf("c");
}
else

{
 if(b>c)
printf("b");
else
printf ("c");
}
```

37: What do you know about else if ladder ?

Ans: For fulfilling if's condition many else if condition are used in a program then it is called else if ladder. An example given below:

```
if(marks>79)
grade= "honours";
else if(marks>59)
grade= "First Division"
else if(marks>49)
grade= "Second Division"
else
grade= "Third Division"
```

38: What do you know about switch statement, give an example?

Ans: Switch statement is used for select one from many paths. Switch(value), when value is matched with case then the case is executed. An example is given below:

```
Index= marks/10;
```

```
Switch (index)
{
 case 10:
case 9:
case 8:
grade= “A+” ;
break;
case 7:
grade= “A” ;
break;
case 6:
grade= “A-” ;
break;
default:
grade= “F” ;
break;
}
```

39: State the following are true or false.

a) When if statements are nested, the last else gets associated with the nearest if without an else. (
b) One if can have more than one else clause. (false)
c) A switch statement can always be replaced by a series of if-else statement. (true)
d) A switch statement can be nay type. (true)
e) A program stops its execution when a break statement is encountered. (true)
f) Each expression in the else if must test the some variable. (true)
g) Each case label can have only one statement. (true)
h) The default case is required in the switch statement. (true)
i) The predicate !(x>=10||y= =5) is equivalent to (x<10&&y!=5). (true)

40: Find errors, if any, in each of the following segment.

a)
```
if (x + y = z && y > 0)
printf(“ ”);
```
b)
```
if (code > 1);
a = b + c
else
```

a = 0

c) if (p<0) | | (q<0)
printf("string is negative");

Ans:

a) if (x + y == z && y > 0)
printf(" ");

b) if (code > 1) [After if statement **;** (semicolon) cannot be used]
a = b + c;
else
a = 0;

c) if (p<0 | | q<0)
printf("string is negative");

41: What will be the values of x and y if n assumes a value of a) 1 and b) 0 ?

```
x = 1;
y = 1;
if (n>0)
x = x +1;
y = y -1;
printf("%d %d", x, y);
```

Output:

When n = 1 than
2 0

When n = 0 than
1 0

42: Find errors which given below.

a) Switch (y);
b) Case 10;
c) Switch (x + y)
d) Switch (x)

```
{
 case 2:
 y = x + y;
break
```

};

Ans:

a) Switch (y)
b) Case 10:
c) Switch (x + y)
d) Switch (x)

```
{
 case 2:
 y = x + y;
break;
}
```

* ***Decision making and looping*** *

43: How many steps are including in a looping process.

Ans: There are four steps are including in a looping process. They are given below:

1. Setting and initialization of a condition variable.
2. Execution of the statements in the loop.
3. Test for a specified value of the condition variable for execution of the loop.
4. Incrementing the condition variable.

44: Explain the while statement with an example.

Ans: The general form of while statement if given below:

```
while(test condition)
{
body of the loop;
}
```

Example:

```
i= 1;
while(i<=5)
{
sum = sum + i;
i + +;
}
```

45: Explain the do while statement with an example.

Ans: The general form of while statement is given below:

```
do
{
 body of the loop;
}
while(test condition);
```

Example:

```
i= 1;
do
{
sum = sum + i;
i + +;
}
while(i<=5);
```

46: Explain the for loop statement with an example.

Ans: The general form of for loop statement is given below:

```
for(initialization; test condition; increment/decrement)
{
body of the loop;
}
```

Example:

```
for(i = 1; i <= 5; i + +)
{
 sum = sum + i;
}
```

47: Explain the nesting of for loops statement with an example.

Ans: When a for loop statement is used into a for loop statement then it is called nesting for loop statement. Here an example is given below:

```
for(i = 1; i<= 5; i + +)
{
for(j = 1; j<=i; j + +)
{
printf(" % d", i);
}
}
```

48: What do you know about continue statement. (Book-171 page)
Ans: The continue statement is the opposite statement of break statement. It represents the loop again. Whatever skips in the loop it starts again testing with the test condition. In while and do-while loops the continue statement directly go to test condition and then continue the looping process. For any for loops it performs first increment part then the conditional test is executes and continuing looping process.

49: State where the following statement are true or false.

a) The do-while statement first executes the loop body and then evaluate the loop control expression. (true)
b) In a pretest loop, if the body is executed n times, the test expression is executed n + 1 times. (true)
c) The number of times a control variable is updated always equals the number of loop iterations. (false)
d) Both the pretest loops include initialization within the statement. (false)
e) In a for loop expression, the starting value of the control variable must be less than its ending value.
f) The initialization, test condition and increment parts may be missing in a for statement. (true)
g) While loop can be used to replace for loops without any change in the body of the loop. (false)

**** Arrays ***

50: What is arrays ?
Ans: An array is a fixed size sequenced collection of elements of the same data type. In its simplest form, an array cane be used to represent a list of numbers, or a list of names.

51: What is the important of an array?
Ans: The important of an array is given below:
When we make a big program, we may need many same type variables. Like (int a, b,--- n) this more variable handling is so difficult. But we can replace this more variable by one array variable with a size like (int x[10])

where we can use 10 int type variable by name of variable x. And the handling of array variable is so easy. Some example where array variable is need to use,

- List of temperature recorded every hour in a day, or month, or year.
- List of employees in an organization.
- List of products and their cost sold by a store.
- Test scores of a class of student.
- Table of daily rainfall data etc.

52: Explain the initialization of an array.
Ans: We can initialization the elements of arrays in the same way as the ordinary variables when they are declaration. The general form of array initialization is given below:
Type array-name [size] = {list values};
The values in the list are separated by commas. For example, the statement
int number [3] = {0,0,0};
Will declare a variable number as an array of size 3 and will assign zero to each element. If the number of values in the list is less than the number of elements, than only that many elements will be initialized. The remaining elements will be set to zero automatically.

53: Explain the initialization of two dimensional arrays.
Ans: Like one dimensional arrays, two dimensional arrays may be initialized by following their declaration with a list of initial values enclosed in braces. For example
int table [2] [3] = {0,0,1,1,1};
Initializes the element of the first row to zero and the second row to one. The initialization is done row by row. The above statement can be equivalently written as
int table [2] [3] = {{0,0},{1,1,1};

54: State where the following statement are true or false.

a) The type pf all elements in an array must be the same. (true)
b) When an array is declared, C automatically initializes its elements to zero. (true)
c) Accessing an array outside its range is a compile time error. (false)
d) A char type variable cannot be used as a subscript in an array. (false)

e) An unsigned long int type cane be used as a subscript. (true)

**** Character Arrays and Strings ****

55: What is the use of getchar and gets function? (Book-234,235 page)
Ans: We can use getchar function repeatedly to read successive single character from the input and place them into a character array. We can use gets function repeatedly to read successive line of a text from the input and place them into an array. The reading terminated when the new line character is entered and null character i.e, then inserted at the end of the string.

56: What is the use of putcahr and puts function ? (Book-240 page)
Ans: Like getchar, C supports another character handling function putchar to output the values of character variable. Example:
char ch = 'A'
putchar (ch);
Like gets, C supports another character handling function puts to output a line of text. Example:
char ch[10] = " Dhaka"
puts (ch);

57: What is the use of strlen, strcpy, strcat, strcmp function ?
Ans: This function counts and returns the number of characters in a string. It takes the following from :
N = strlen (string);
This strcpy function works almost like a string assignment operator. It takes the form
Strcpy (string1,string2); and assigns the content of string2 to string1.
The strcat function joins two strings together. It takes form
Strcat (string1,string2);
The strcmp function compares two strings . It takes the form
Strcmp (string1, string2)

**** User-Defined Function ****

58: What are the elements of the function?

Ans: The elements of a function are given below:

1) Function name
2) Function type
3) List of parameter.
4) Local variable
5) Function statements
6) A return statement.

59: Write the types categories of a function with an example.

Ans: There are five categories of function that are given below:

1) Function with no arguments and no return values.

Example: add ( );

2) Function with arguments and no return values.

Example: add (x);

3) Function with arguments and return values.

Example: int add (x);

4) Function with no arguments but with one return value.

Example: int add ( );

5) Function that return multiple values.

60: What is recursion give an example?

Ans: (**By Subroto sir:** The process in which a function in turn calls itself is called recursion. And the function is called recursive function.**)**

When a called function in turn calls another function again and again the process of this calling is called recursion. An example is given below:

```
#include<stdio.h>
#include<conio.h>
int fact(int n);
void main()
        {
         int n, f;
         clrscr();
```

```
printf ("input the value of n :");
scanf ("%d", &n);
f = fact (n);
printf ("factorial = %d", f);
getch();
}
int fact(int n)
{
int p;
if(n = = 1)
return 1;
else
p  = n * fact(n-1);
return p;
}
```

61: State where the following statement are true or false.

a) C function can return only one value under their function name. (true)
b) A function in C should have at lest one argument. (false)
c) A function cane be defined and placed before the main. (true)
d) A function can be defined within the main function. (false)
e) Any name cane be used as a function name. (true)
f) Only a void type function can have void as its argument. (false)
g) A function can call itself. (true)

**** Pointer ***

62: Explain the declaration of pointer variables.

Ans: In C, every variable must be declared for its type. Since pointer variables contain addresses that belong to a separate data type, they must be declared as pointers before we use them. The declaration of a pointer variable takes the form:

Data type *pt name;

63: Explain the initialization of pointer variables.

Ans: the process of assigning the address of a variable to a pointer variable to a pointer variable is known as initialization. As pointed out earlier, all

initialized pointers will have some unknown values that will be interpreted as memory addresses. They may not be valid addresses or they may point to some values that are wrong. Since the compilers do not be valid addresses or they may point some values that are wrong. Since the compilers do not detect these errors, the programs with initialized pointers will produce erroneous results. It is therefore important to initialize pointer variable carefully before they are used in the program.
Once a pointer variable has been declared we can use the assignment operator to initialize the variable. Example:
Int x, *p;
P = &x;

64: What do you know about call by reference and call by value?
Ans: The process of calling a function using pointers to pass the addresses of variables is known as "call by reference". And it is given below:

```
main ( )
{
int x, y;
printf(" input the value of x & y :");
scanf(" %d %d", &x , &y);
add (&x, &y);      /*this is called by reference*/
getch ();
}
```

The process of calling function by the actual value of a variable is known as "call by value". An example is given below:

```
main ( )
{
int x, y;
printf (" input the value of x & y:");
scanf (" %d %d", &x ,&y);
add (x, y);   /*this is called by value*/
getch ();
}
```