10th Class		
Computer Science	Model Paper 7	Paper: II
Time: 1.45 Hours	(Subjective Type)	Marks: 40

(Part-I)

2. Write short answers to any FOUR (4) questions: (8)

(i) Why we need IDE?

Ans We need IDE because it consists of tools that help a programmer throughout the phases of writing, executing and testing a computer program.

(ii) What is difference between Real Constants and Character Constants?

Ans

Real Constants	Character Constants
Real constants are the values including a decimal point. They can be positive or negative.	Any single small case letter, uppercase letter, digit, punctuation mark, special symbol enclosed within ' ' is considered a character constant.
Example: 3.14, 15.3333, 75.0, 1575.76, 7941.2345, etc.	**Example:** '5' '7', 'a', 'X', '!', etc.

(iii) What is formation of escape sequence?

Ans Escape sequence consists of two characters:

1. The first character is always back slash (\).
2. The second character varies according to the functionality that we want to achieve.

Back slash (\) is called escape character which is associated with each escape sequence to notify about escape. Escape character and character next to it are not displayed on screen, but they perform specific task assigned to them.

(iv) Can we display output on multiple line with multiple printf statement in the absence of escape sequence?

Ans In the absence of an escape sequence, even if we have multiple printf statement, their output is displayed on a single line. Following example illustrates this point.

Example:

```
#include<stdio.h>
void main()
{
        printf("My name is");
        printf("Ahmad");
}
```

Output:

My name is Ahmad.

(v) What is difference between multiplication operator and addition operator?

Ans

Multiplication operator	Addition operator
Multiplication operator (*) is a binary operator which performs the product of two numbers. It is denoted by *.	Addition operator calculates the sum of two operands. It is denoted by (+).

(vi) Which operator is applied on three operands?

Ans C language also offers a ternary operator that is applied on three operands.

3. Write short answers to any FOUR (4) questions: (8)

(i) Define associated code.

Ans The associated code in if statement is any valid C-language set of statements.

(ii) Write name of selection statements types.

Ans Two types of selection statements are:

1. If statement
2. If-else statement

(iii) Define block or compound statement.

Ans A set of multiple instructions enclosed in braces is called a block or a compound statement.

(iv) Differentiate between sequential and selection statements.

Ans

Sequential Control	Selection Statements
Sequential control is the default control structure in C language. According to the sequential control, all the statements are executed in the given sequence.	The statements which help us to decide which statements should be executed next, on the basis of conditions, are called selection statements.

(v) What is Data Type?

Ans Data type is the type of data that we want to store in the array.

(vi) Write down the method to initialize an array.

Ans An array can be initialized at the time of its declaration, or later. Array initialization at the time of declaration can be done in the following manner:

data_type array_name [N] = {value1, value2, value3, ..., valueN};

4. Write short answers to any FOUR (4) questions: (8)

(i) How to access an array element?

Ans Each element of an array has an index that can be used with the array name as array_name [index] to access the data stored at that particular index.

First element has the index 0, second element has the index 1 and so on. Thus height[0] refers to the first element of array height, height[1] refers to the second element and so on.

(ii) What is difference between arguments and parameters?

Ans The values passed to the function are called arguments, whereas variables in the function definition that receive these values are called parameters of the function.

(iii) Can there are multiple return statements in a function?

Ans There may be multiple return statements in a function but as soon as the first return statement is executed, the function call returns and further statements in the body of function are not executed.

(iv) Write down the output of following code.

```
main()
{
    int x = 20;
```

```
        int y = 10;
        swap(x,y);
        printf("%d %d", y, x + 2);
}
        swap(int x,int y)
{
        int temp;
        temp = x;
        x = y;
        y = temp;
}
```

Ans **Output:**

10 22 (duplicate copy of arguments are generated on calling a function).

(v) **Identify the error from the following code.**

```
int max (int a; int b)
{
    if (a > b)
    return a;
    return b;
}
```

Ans **Errors:**

Use comma instead of semicolon between parameters.

(vi) **How can we handle the complexity of the problem?**

Ans If we write the whole program as a single procedure, management of the program becomes difficult. Functions divide the program into smaller units, and thus reduce the complexity of the problem.

(Part-II)

NOTE: Attempt any TWO (2) questions.

Q.5. What are the operators? Describe assignment operators and arithmetic operators in details. (8)

Ans **Operators:**

The name computer suggests that computation is the most important aspect of computers. We need to perform computations on data through programming. We have a lot of

mathematical functions to perform calculations on data. We can also perform mathematical operations in our programs. C language offers numerous operators to manipulate and process data. Following is the list of some basic operator types:

- Assignment operator
- Arithmetic operators
- Logical operators
- Relational operators

Assignment Operator:

Assignment operator is used to assign a value to a variable, or assign a value of variable to another variable.

Equal sign (=) is used as assignment operator in C. Consider the following example:

```
int sum = 5;
```

Value 5 is assigned to a variable named *sum* after executing this line of code. Let's have a look at another example:

```
int sum = 6;
int var  = sum;
```

First, value 6 is assigned to variable *sum*. In the next line, the value of *sum* is assigned to variable *var*.

Arithmetic Operators:

Arithmetic operators are used to perform arithmetic operations on data. Following table represents arithmetic operators with their description.

Operators	Name	Description
/	Division Operator	It is used to divide the value on left side by the value on right side.
*	Multiplication Operator	It is used to multiply two values.
+	Addition Operator	It is used to add two values.
–	Subtraction Operator	It is used to subtract the value on right side from the value on left side.
%	Modulus Operator	It gives remainder value after dividing the left operand by right operand.

Table: Arithmetic Operators.

Division:

Division operator (/) divides the value of left operand by the value of right operand. *e.g.*, look at the following statement.

float result = 3.0 / 2.0;

After this statement, the variable result contains the value 1.5.

Multiplication:

Multiplication operator (*) is a binary operator which performs the product of two numbers. Look at the following statement:

```
int multiply = 5 * 5;
```

After the execution of statement, the variable multiply contains value 25.

Addition:

Addition operator (+) calculates the sum of two operands. Let's look at the following statement:

```
int add = 10 + 10;
```

Resultant value in variable *add* is 20.

Subtraction:

Subtraction operator (–) subtracts right operand from the left operand. Let's look at the following statement:

```
int result = 20 – 15;
```

After performing subtraction, value 5 is assigned to the variable *result*.

Modulus operator:

Modulus operator (%) performs division of left operand by the right operand and returns the remainder value after division. Modulus operator works on integer data types.

```
int remaining = 14 % 3;
```

As, when we divide 14 by 3, we get a remainder of 2, so the value stored in variable remaining is 2.

Q.6. Write a program that takes the type of consumer and number of units consumed as input. The program then displays the electricity bill of the user. (8)

Ans

```
#include<stdio.h>
void main()
{
    int units, unit_price, bill;
    char user_type;
    printf("Please enter h for home user and c for
    commercial user:");
    scanf("%c", & user_type);
```

```
printf("Please enter the number of units consumed:");
scanf("%d", &units);
if(units <= 200)
    if(user_type = = 'h')
            unit_price = 12;
    else if (user_type = = 'c')
            unit_price = 15;
else if (units > 200 && units <= 400)
    if(user_type = = 'h')
            unit_price = 15;
    else if (user_type = = 'c')
            unit_price = 20;
else
    if(user_type = = 'h')
            unit_price = 15;
    else if (user_type = = 'c')
            unit_price = 24;
bill = units * unit_price;
printf("Your electricity bill is %d", bill);
}
```

Q.7. What is loop structure? Describe the structure of a for loop. (8)

Ans **Loop Structure:**

If we need to repeat one or more statements, then we use loops. For example, if we need to write Pakistan thousand times on the screen, then instead of writing printf("Pakistan"); a thousand times, we use loops. C language provides three kind of loop structures:

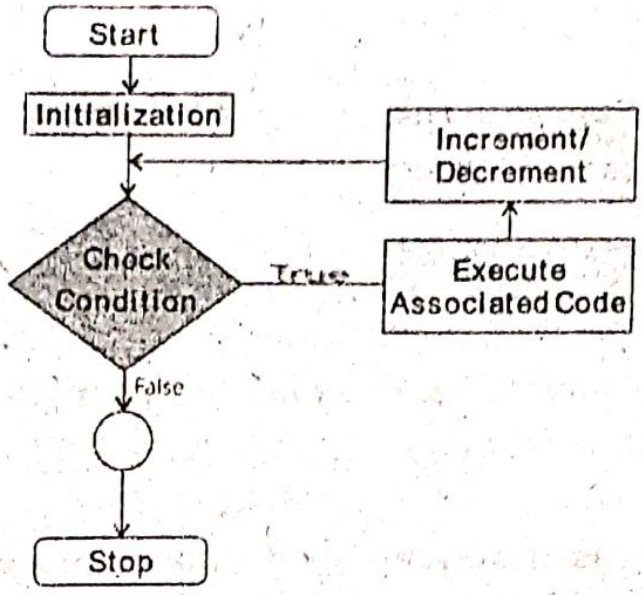
1- For loop
2- While loop
3- Do While loop

General syntax of for loop:

In C programming language, for loop has the following general syntax:

```
for(initialization; condition; increment/decrement)
{
    Code to repeat
}
```

In order to understand the *for* loop structure, let's look at the following flow chart:



From the flow chart, we can observe the following sequence:

1. *Initialization* is the first part to be executed in a *for* loop. Here we initialize our counter variable and then move to the *condition* part.
2. *Condition* is checked, and if it turns out to be *false*, then we come out of loop.
3. If the *condition* is *true*, then *body of the loop* is executed.
4. After executing the body of loop, the counter variable is increased or decreased depending on the used logic, and then we again move to the step 2.

After executing the *body of loop*, the counter variable is increased or decreased depending on the used logic, and then we again move to the step 2.

Example:

```
for (int i = 0; i < 3; i++ )
{
    printf("Pakistan\n" );
}
```

Output:

```
Pakistan
Pakistan
Pakistan
```