BSEK • 10TH CLASS • KARACHI BOARD

10th Class Computer Science Notes

Full-Book Notes, MCQs and Solved Questions
Board of Secondary Education Karachi (BSEK) — English

Unit-wise revision notes, solved multiple-choice questions,
short questions and long questions — written in original words for these notes.

These notes follow the Sindh Textbook Board book
The Textbook of Computer Science for Grade 10 (Test Edition, SELD/STBB-18/2021),
Units 1 to 7. Check the unit titles against your own issued book,
because boards may revise them.

Prepared for students of 10th Class Computer Science (BSEK Karachi). All practice questions and code examples in this book are original and written for these notes.

Contents

1. Unit 1 — Problem Solving and Algorithm Designing
2. Unit 2 — Basics of Programming in C++
3. Unit 3 — Input/Output Handling in C++
4. Unit 4 — Control Structure
5. Unit 5 — Functions
6. Unit 6 — Digital Logic and Design
7. Unit 7 — Introduction to Scratch

- **MCQ Answer Key**

How to Use This Book

- Read the revision notes of one unit first and underline the key terms.
- Then attempt the 10 MCQs of that unit without looking at the answer key, and check your answers at the end of the book.
- Write the short and long questions in your own words after reading the given answers.
- Type each C++ code example into a compiler, run it, and change the values to see what happens — typing the code yourself is the best practice.
- For Unit 7, open Scratch and rebuild the block scripts described in the notes, step by step.
- Revise the key terms of every unit before the examination.

A Note on the Examination Paper

The Class 10 Computer Studies theory paper in Sindh is 60 marks, 3 hours — Section A: 12 MCQs (12 marks), Section B: 8 short questions out of 12 (24 marks), Section C: 4 descriptive questions out of 6 (24 marks) — as in the BSEK model papers (2023 and 2024). A practical examination is also held; its marks are not printed here because published sources disagree about them. Students should check the current model paper on the official BSEK website, bsek.edu.pk, before the examination.

Code Examples and Scratch Scripts

Each programming unit ends with a short, original C++ program, and Unit 7 describes Scratch block scripts in words. The programs are written for these notes and are not copied from the book — type them, run them, and try your own changes.

All practice questions in this book are original and written for these notes. They are not copied from any past paper or guidebook.

Unit 1: Problem Solving and Algorithm Designing

What Is Problem Solving?

Problem solving is the process of finding a clear, step-by-step solution to a problem before writing a computer program. A good programmer first understands the problem, then plans the solution, then writes the program, and finally tests it. The usual steps are: understand the problem, plan a solution, design an algorithm, write the program, test and debug it, and document and maintain it.

What Is an Algorithm?

An algorithm is a finite, step-by-step procedure for solving a problem. Every step must be clear and exact, so that anyone following the steps reaches the same correct answer. For example, the algorithm for making tea is: boil water, put tea leaves in the pot, pour the boiling water, wait for a few minutes, and serve. In computing, an algorithm is written before the program, because a clear algorithm makes the program easy to write and easy to check.

Characteristics of a Good Algorithm

- Input: an algorithm takes zero or more values as input.
- Output: it produces at least one result as output.
- Definiteness: every step is clear and unambiguous.
- Finiteness: it finishes after a finite number of steps.
- Effectiveness: every step is simple enough to be carried out.

Flowcharts

A flowchart is a diagram that shows the steps of an algorithm using standard symbols connected by arrows. It helps the programmer to see the logic of a solution at a glance. The main symbols are: the oval (terminal) for START and STOP; the parallelogram for input and output; the rectangle (process box) for calculations and assignments; the diamond for a decision, with two arrows leaving it, YES and NO; and small circles for connectors. Arrows show the direction of flow.

Pseudocode

Pseudocode is a way of writing an algorithm in plain, English-like statements instead of a diagram. It is not a real programming language, so it has no strict rules, but it follows the same order of steps as the final program. Pseudocode is easier to write than a flowchart for long problems, and it is easy to turn into C++ code afterwards.

Algorithm Design: Sequence, Selection and Repetition

Every algorithm is built from three basic structures. Sequence means steps carried out one after another, in order. Selection means choosing between two or more paths, for example with an IF-THEN-ELSE decision. Repetition means repeating a group of steps, for example with a loop, until a condition is met. Any problem can be solved by combining these three structures.

Dry Run (Trace Table)

A dry run is a hand test of an algorithm. The programmer takes a few sample values, follows the steps one by one on paper, and writes the changing values in a trace table. A dry run finds mistakes in the logic before the program is written, so it saves time in testing later.

Code Example — Pseudocode: Largest of Three Numbers

```
START
  INPUT A, B, C
  IF A > B AND A > C THEN
```

```
      PRINT "Largest is A"
  ELSE IF B > A AND B > C THEN
      PRINT "Largest is B"
  ELSE
      PRINT "Largest is C"
  ENDIF
STOP
```

Unit 1: Problem Solving and Algorithm Designing — Practice Questions

Multiple Choice Questions

1. An algorithm is

(a) a step-by-step procedure to solve a problem (b) a flowchart symbol (c) a programming language (d) a computer program

2. Which of these is a characteristic of a good algorithm?

(a) It has unclear steps (b) It finishes after a finite number of steps (c) It has no output (d) It never ends

3. Definiteness in an algorithm means

(a) it has no output (b) it takes input (c) every step is clear and unambiguous (d) it runs forever

4. The diamond symbol in a flowchart is used for

(a) input and output (b) a process (c) start and stop (d) a decision

5. The parallelogram symbol in a flowchart is used for

(a) input and output (b) a process (c) start and stop (d) a decision

6. The rectangle (process box) in a flowchart is used for

(a) input and output (b) calculations and assignments (c) start and stop (d) a decision

7. Pseudocode is

(a) a computer program (b) a real programming language (c) an algorithm written in plain English-like statements (d) a type of flowchart symbol

8. The three basic structures of an algorithm are

(a) start, stop and process (b) if, else and switch (c) input, output and storage (d) sequence, selection and repetition

9. A dry run is

(a) testing an algorithm by hand on paper (b) printing a flowchart (c) writing pseudocode (d) running a program on the computer

10. The first step in problem solving is to

(a) test the program (b) understand the problem (c) draw the flowchart (d) write the program

Short Questions

1. Define an algorithm.

Answer: An algorithm is a finite, step-by-step procedure for solving a problem, in which every step is clear and exact so that following the steps always gives the same correct answer.

2. Name any two characteristics of a good algorithm.

Answer: Two characteristics of a good algorithm are definiteness — every step is clear and unambiguous — and finiteness — the algorithm finishes after a finite number of steps.

3. Which flowchart symbol is used for a decision?

Answer: The diamond symbol is used for a decision in a flowchart; two arrows leave it, one for YES and one for NO.

4. What is pseudocode?

Answer: Pseudocode is a way of writing an algorithm in plain, English-like statements. It is not a real programming language, so it has no strict rules, but it follows the same order of steps as the final program.

5. What is a dry run?

Answer: A dry run is a hand test of an algorithm: the programmer takes sample values, follows the steps one by one on paper, and records the changing values in a trace table to find mistakes in the logic.

Long Questions

1. Describe the steps involved in solving a problem with a computer.

Answer: Problem solving has six steps. First, understand the problem: read it carefully and find what is given and what is required. Second, plan a solution: decide the method and the inputs, processes and outputs. Third, design an algorithm: write the steps in order, as a flowchart or in pseudocode. Fourth, write the program: turn the algorithm into code in a programming language such as C++. Fifth, test and debug: run the program with sample data, find the errors and correct them. Sixth, document and maintain: write notes about the program and update it when needed.

2. Write an algorithm to find the average of three numbers and explain its steps.

Answer: The algorithm is: START; INPUT A, B, C; SET Sum = A + B + C; SET Average = Sum / 3; PRINT Average; STOP. The steps are: first take three numbers as input, then add them to get their sum, then divide the sum by 3 to get the average, and finally print the average as the output. The algorithm has input (A, B, C), a clear process (addition and division), and output (Average), and it finishes in a finite number of steps.

Unit 2: Basics of Programming in C++

A Brief History of C++

C++ was developed by Bjarne Stroustrup at Bell Laboratories. In 1979 he created a language called 'C with Classes', adding classes to the C language, and in 1983 it was renamed C++. C++ is a general-purpose, middle-level language: it has the power of a low-level language and the features of a high-level, object-oriented language. It is used for operating systems, games, and large software.

Structure of a C++ Program

A simple C++ program has these parts: preprocessor directives such as #include <iostream>, which bring in library files; the main() function, where the program begins; a pair of braces { } that mark the body of the function; statements, each ending with a semicolon; and a return 0; statement that ends the program. The line using namespace std; lets the program use cin and cout without the std:: prefix.

Data Types

- int: whole numbers, for example 5, -20, 1000 (commonly 4 bytes).
- float: numbers with a decimal point, for example 3.14 (commonly 4 bytes).
- double: decimal numbers with more precision than float (commonly 8 bytes).
- char: a single character, for example 'A' or '7' (1 byte).
- bool: true or false (1 byte).
- void: no value; used as the return type of functions that return nothing.

Variables and Constants

A variable is a named place in memory that stores a value which can change while the program runs. It must be declared with a data type before use, for example int marks; and it can be given a value at once, for example int marks = 85;. A constant is a value that cannot change; it is declared with the const keyword, for example const float PI = 3.14159;.

Identifiers

An identifier is the name given to a variable, function or other item in a program. An identifier must start with a letter or an underscore, and may contain letters, digits and underscores. It must not be a keyword, and it must not contain a space or a special character. Identifiers are case-sensitive, so marks and Marks are two different names.

Keywords

Keywords are words reserved by the language for a special purpose, so they cannot be used as identifiers. Common C++ keywords are int, float, double, char, bool, void, if, else, for, while, do, switch, case, break, continue, return, const and class.

Operators

- Arithmetic: + (add), - (subtract), * (multiply), / (divide), % (remainder).
- Relational: == (equal to), != (not equal to), < (less than), > (greater than), <= (less than or equal to), >= (greater than or equal to).
- Logical: && (AND), || (OR), ! (NOT).
- Assignment: = (assign), +=, -=, *=, /= (assign after the operation).
- Increment and decrement: ++ (add 1), -- (subtract 1).

Input and Output

C++ uses the objects cout and cin from the iostream library. cout with the insertion operator << sends output to the screen, for example cout << "Hello";. cin with the extraction operator >> reads input from the keyboard, for example cin >> marks;. The using namespace std; line lets the program write cout and cin instead of std::cout and std::cin.

Comments

A comment is a note for the reader that the compiler ignores. A single-line comment starts with // and runs to the end of the line. A multi-line comment starts with /* and ends with */. Comments explain what the code does and make programs easier to read.

Code Example — Adding Two Numbers

```
#include <iostream>
using namespace std;

int main()
{
    int a, b, sum;          // declare three integer variables
    cout << "Enter two numbers: ";
    cin >> a >> b;          // read two numbers from the keyboard
    sum = a + b;            // add them
    cout << "Sum = " << sum;
    return 0;
}
```

Unit 2: Basics of Programming in C++ — Practice Questions

Multiple Choice Questions

1. C++ was developed by

(a) Charles Babbage (b) Dennis Ritchie (c) Bjarne Stroustrup (d) James Gosling

2. C++ was first called

(a) C Sharp (b) Objective C (c) Visual C (d) C with Classes

3. The header file used for cin and cout is

(a) iostream (b) string.h (c) math.h (d) stdio.h

4. Every C++ statement ends with a

(a) colon (b) semicolon (c) full stop (d) comma

5. The main() function returns a value of type

(a) float (b) char (c) int (d) void

6. Which data type is used for decimal numbers?

(a) int (b) char (c) bool (d) float

7. Which of these is a keyword in C++?

(a) if (b) result (c) marks (d) total

8. A single-line comment in C++ starts with

(a) # (b) // (c) /* (d) <!--

9. The extraction operator used with cin is

(a) && (b) << (c) >> (d) ==

10. Which of these is NOT a valid identifier?

(a) value2 (b) my_marks (c) _total (d) 2value

Short Questions

1. Define a variable.

Answer: A variable is a named place in memory that stores a value which can change while the program runs; it must be declared with a data type before use, for example int marks;.

2. What is the difference between float and double?

Answer: Both float and double store decimal numbers, but double stores them with more precision and commonly uses 8 bytes, while float commonly uses 4 bytes, so double is used when greater accuracy is needed.

3. What does #include <iostream> do?

Answer: The line #include <iostream> brings the iostream library into the program, so that the input and output objects cin and cout can be used.

4. What is a keyword?

Answer: A keyword is a word reserved by the C++ language for a special purpose, such as int, if, for or return, and it cannot be used as the name of a variable or function.

5. Why is a semicolon used in C++?

Answer: A semicolon marks the end of a statement in C++, so the compiler knows where one statement ends and the next begins.

Long Questions

1. Explain the structure of a C++ program with the help of an example.

Answer: A C++ program begins with preprocessor directives such as #include <iostream>, which bring in library files, and using namespace std;, which lets the program use cin and cout. Then comes the main() function, where the program starts; its body is written between braces { }. Inside main, statements are written, each ending with a semicolon, and return 0; ends the program. For example, a program to add two numbers declares int a, b, sum;, reads a and b with cin, adds them with sum = a + b;, and prints the result with cout. Every part has a fixed place, and missing any part causes an error.

2. Explain the data types of C++ with examples.

Answer: C++ has several built-in data types. int stores whole numbers, for example int age = 16;. float stores decimal numbers, for example float price = 99.50;. double stores decimal numbers with more precision, for example double pi = 3.14159265;. char stores a single character in single quotes, for example char grade = 'A';. bool stores true or false, for example bool pass = true;. void means no value and is used as the return type of functions that return nothing. The programmer chooses the type that fits the value being stored.

Unit 3: Input/Output Handling in C++

The cin Object

The cin object reads input from the keyboard. It is used with the extraction operator >>, for example cin >> age;. More than one value can be read in one statement, for example cin >> a >> b;, which reads two values separated by a space or by pressing Enter. cin stops reading at a space, so it reads only one word at a time.

The cout Object

The cout object sends output to the screen. It is used with the insertion operator <<, for example cout << "Result = " << result;. Several items can be printed in one statement by chaining them with <<, and text is written inside double quotes while variables are written without quotes.

Reading Strings

The cin object stops at the first space, so cin >> name; reads only the first word of a name such as 'Ali Raza'. To read a whole line, including spaces, the getline() function is used, for example getline(cin, name);. A C-style string is an array of characters ending with a null character '\0'.

Escape Sequences

- \n: moves to a new line.
- \t: moves to the next tab position (a horizontal tab).
- \\: prints a backslash.
- \": prints a double quote inside a string.
- \': prints a single quote.

Formatted Output

Output can be formatted to look neat. The manipulator endl moves to a new line and flushes the output. The manipulator setw(n), from the iomanip library, sets the width of the next output field. The manipulator setprecision(n), also from iomanip, sets the number of significant digits shown, and fixed makes setprecision count the digits after the decimal point. For example, cout << fixed << setprecision(2) << 12.3456; prints 12.35.

Type Conversion

Type conversion changes a value from one data type to another. Implicit conversion happens automatically, for example when an int value is stored in a float variable. Explicit conversion, called type casting, is done by the programmer, for example float result = (float) 7 / 2;, which gives 3.5 instead of 3, or static_cast<float>(7) / 2. Casting is needed when integer division would otherwise lose the decimal part.

Code Example — Reading a Name and Marks

```
#include <iostream>
#include <iomanip>   // for setw and setprecision
#include <string>
using namespace std;

int main()
{
    string name;
    int marks;
    cout << "Enter your name: ";
    getline(cin, name);        // reads the full name, spaces included
```

```
    cout << "Enter your marks: ";
    cin >> marks;
    cout << "Name: " << name << endl;
    cout << "Marks: " << setw(5) << marks << endl;
    cout << "Percentage: " << fixed << setprecision(2)
         << (float)marks << "%" << endl;
    return 0;
}
```

Unit 3: Input/Output Handling in C++ — Practice Questions

Multiple Choice Questions

1. The extraction operator used with cin is

(a) >> (b) == (c) && (d) <<

2. The insertion operator used with cout is

(a) != (b) << (c) >> (d) ==

3. cin stops reading input at a

(a) comma (b) full stop (c) space (d) semicolon

4. To read a whole line including spaces, we use

(a) cout << (b) scanf() (c) cin >> (d) getline()

5. The escape sequence \n is used to

(a) move to a new line (b) print a backslash (c) end the program (d) print a tab

6. The escape sequence \t is used to

(a) print a quote (b) insert a tab (c) print a backslash (d) move to a new line

7. The manipulator setw() is defined in the header file

(a) cmath (b) iostream (c) iomanip (d) string

8. The manipulator endl is used to

(a) clear the screen (b) read input (c) end the program (d) move to a new line

9. setprecision() with fixed controls

(a) the digits after the decimal point (b) the size of a variable (c) the number of lines (d) the width of the output

10. Changing a value from one data type to another by the programmer is called

(a) type reading (b) type casting (c) type naming (d) type printing

Short Questions

1. What is the difference between cin and cout?

Answer: cin is used to read input from the keyboard with the extraction operator >>, while cout is used to send output to the screen with the insertion operator <<.

2. What does getline() do?

Answer: getline() reads a whole line of text from the keyboard, including spaces, into a string variable, for example getline(cin, name);, whereas cin >> stops at the first space.

3. Name two escape sequences and their use.

Answer: \n moves the output to a new line, and \t moves it to the next tab position. Other escape sequences are \\ for a backslash and \" for a double quote.

4. What is the use of endl?

Answer: The manipulator endl moves the output to a new line and flushes the output buffer, so the text appears on the screen at once.

5. What is type conversion?

Answer: Type conversion changes a value from one data type to another. It is implicit when it happens automatically, and explicit, or type casting, when the programmer writes it, for example (float) 7 / 2 gives 3.5 instead of 3.

Long Questions

1. Explain input and output handling in C++ with examples.

Answer: Input in C++ is handled by the cin object with the extraction operator >>, for example cin >> age; reads one value, and cin >> a >> b; reads two values. Output is handled by the cout object with the insertion operator <<, for example cout << "Age = " << age; prints text and a variable. To read a whole line with spaces, getline(cin, name); is used, because cin stops at a space. Escape sequences such as \n for a new line and \t for a tab control the layout of the output, and the manipulator endl moves to a new line. Together, cin and cout let a program take data from the user and show results on the screen.

2. Explain formatted output in C++ using manipulators, with a program.

Answer: Formatted output is output arranged neatly with manipulators from the iomanip library. setw(n) sets the width of the next output field, for example setw(5) prints a number in a field 5 characters wide. setprecision(n) sets the number of significant digits, and fixed makes it count the digits after the decimal point, so cout << fixed << setprecision(2) << 12.3456; prints 12.35. endl moves to a new line. A program that reads a student's name and marks can use getline(cin, name); to read the name, then print the marks with setw(5) and the percentage with fixed << setprecision(2), so the result is shown in neat columns with two decimal places.

Unit 4: Control Structure

What Is a Control Structure?

A control structure changes the order in which the statements of a program are carried out. Without control structures, statements run one after another from top to bottom, which is called sequence. Selection structures choose between paths, and repetition structures repeat a group of statements. C++ provides if, if-else and switch for selection, and for, while and do-while for repetition.

The if Statement

The if statement runs a block of code only when its condition is true. The condition is written in parentheses and uses a relational or logical operator, for example if (marks >= 40) { cout << "Pass"; }. If the condition is false, the block is skipped and the program continues after it.

The if-else Statement

The if-else statement chooses between two blocks: the if block runs when the condition is true and the else block runs when it is false, for example if (number % 2 == 0) { cout << "Even"; } else { cout << "Odd"; }. Only one of the two blocks ever runs.

Nested if and the else-if Ladder

An if statement inside another if statement is called a nested if. An else-if ladder tests several conditions one after another, for example to give a grade: if marks are 80 or more, grade A; else if 70 or more, grade B; else if 60 or more, grade C; else, grade D. The first true condition runs its block and the rest of the ladder is skipped.

The switch Statement

The switch statement chooses one of many cases by the value of an expression, which must be an integer or a character. Each case has a value followed by a colon, and the break statement stops the switch after the matching case; without break, the program falls through to the next case. The default case runs when no case matches. switch is a neater choice than a long else-if ladder when one value is being tested against many fixed values.

The for Loop

The for loop repeats a block a known number of times. It has three parts in its header: initialization, which runs once at the start; the condition, which is tested before every repetition; and the increment or decrement, which runs after every repetition. For example, for (int i = 1; i <= 10; i++) { cout << i; } prints the numbers 1 to 10.

The while Loop

The while loop repeats a block while its condition is true, and the condition is tested before every repetition. It is used when the number of repetitions is not known in advance, for example while (number != 0) { sum = sum + number; cin >> number; }. If the condition is false at the start, the loop body never runs.

The do-while Loop

The do-while loop is like the while loop, except that the condition is tested after the body, so the body always runs at least once. For example, do { cout << "Enter a number: "; cin >> number; } while (number < 0); asks for a number and repeats the question while the number is negative. It is useful for menus and for input that must be checked.

break and continue

The break statement ends a loop or a switch at once, and the program continues after it. The continue statement skips the rest of the current repetition and jumps to the next one: in a for loop it runs the increment first, and in a while loop it goes straight to the condition test.

Code Example — Multiplication Table

```
#include <iostream>
using namespace std;

int main()
{
    int number;
    cout << "Enter a number: ";
    cin >> number;
    // print the table of the number from 1 to 10
    for (int i = 1; i <= 10; i++)
    {
        cout << number << " x " << i << " = " << number * i << endl;
    }
    return 0;
}
```

Unit 4: Control Structure — Practice Questions

Multiple Choice Questions

1. Selection structures in C++ are

(a) break and continue (b) for and while (c) if and switch (d) do and while

2. The if statement runs its block only when the condition is

(a) zero (b) negative (c) false (d) true

3. In an if-else statement, how many blocks run?

(a) only one of the two (b) none (c) all of them (d) both

4. The switch statement works with values of type

(a) string only (b) integer or character (c) double only (d) float only

5. The break statement in a switch is used to

(a) end the program (b) skip one case (c) stop the switch after the matching case (d) repeat the switch

6. A for loop has how many parts in its header?

(a) four (b) one (c) two (d) three

7. Which loop always runs its body at least once?

(a) do-while (b) nested loop (c) for (d) while

8. In a while loop the condition is tested

(a) after every repetition (b) before every repetition (c) only once (d) never

9. The continue statement

(a) repeats the loop forever (b) ends the loop (c) skips to the next repetition (d) ends the program

10. A loop inside another loop is called a

(a) infinite loop (b) empty loop (c) do loop (d) nested loop

Short Questions

1. What is the difference between if and if-else?

Answer: An if statement runs its block only when the condition is true and does nothing otherwise, while an if-else statement runs the if block when the condition is true and the else block when it is false, so one of the two blocks always runs.

2. What is a loop?

Answer: A loop is a control structure that repeats a group of statements while a condition holds, so the same code runs many times without being written again, for example a for, while or do-while loop.

3. Why is break used in a switch statement?

Answer: break stops the switch after the matching case has run; without it the program falls through and runs the following cases as well, which is usually not intended.

4. What is the difference between while and do-while?

Answer: In a while loop the condition is tested before the body, so the body may never run, while in a do-while loop the condition is tested after the body, so the body always runs at least once.

5. What is a nested loop?

Answer: A nested loop is a loop written inside another loop; the inner loop runs completely for every single repetition of the outer loop, and nested loops are used for tables, patterns and two-dimensional data.

Long Questions

1. Explain the three loops in C++ with examples.

Answer: C++ has three loops. The for loop is used when the number of repetitions is known: for (int i = 1; i <= 10; i++) prints the numbers 1 to 10, with initialization, condition and increment in its header. The while loop repeats while its condition is true and tests the condition before the body, for example while (number != 0) keeps adding numbers until 0 is entered; it is used when the number of repetitions is not known in advance. The do-while loop tests its condition after the body, so it always runs at least once, for example do { cin >> number; } while (number < 0); keeps asking for a number while it is negative. All three can be controlled with break and continue.

2. Explain the switch statement and write a program that displays the day of the week.

Answer: The switch statement chooses one case by the value of an integer or character expression. Each case has a value and a colon, break stops the switch after the matching case, and default runs when no case matches. A program to display the day reads a number from 1 to 7, then switches on it: case 1 prints Monday, case 2 prints Tuesday, and so on up to case 7 for Sunday, with default printing 'Invalid day'. switch is neater than a long else-if ladder when one value is tested against many fixed values.

Unit 5: Functions

What Is a Function?

A function is a named block of code that performs one task. Instead of writing the same code again and again, the programmer writes it once as a function and calls it whenever it is needed. Functions make a program shorter, easier to read, and easier to test, because each part of the work is kept in its own place. This way of writing programs is called modular programming.

Function Declaration and Definition

A function has a return type, a name, and a list of parameters in parentheses. The declaration, also called the prototype, tells the compiler about the function before it is used, for example int add(int, int);. The definition gives the actual code of the function, with the body written between braces, for example int add(int a, int b) { return a + b; }. The declaration ends with a semicolon; the definition does not.

Calling a Function

A function is used by calling it: writing its name followed by the values it needs in parentheses, for example int total = add(5, 7);. When a function is called, the program jumps to the function, runs its body, and returns to the place of the call with the result. The main() function is itself a function, and every C++ program must have one.

Parameters and Arguments

Parameters are the variables listed in a function's definition; they receive values when the function is called. The values passed in the call are called arguments, or actual parameters. In C++, arguments are normally passed by value: a copy of the value is given to the function, so changing the parameter inside the function does not change the original variable in the caller.

Return Values and void Functions

A function returns a value with the return statement, and the return type must match the type of the value, for example int add(int a, int b) { return a + b; }. A function that returns nothing has the return type void and needs no return statement with a value, for example void showMessage() { cout << "Hello"; }.

Local and Global Variables

A local variable is declared inside a function or a block and can be used only there; it is created when the function runs and destroyed when it ends. A global variable is declared outside all functions and can be used by every function in the program; it exists for the whole run of the program. Local variables are preferred, because global variables can be changed by any function and make errors hard to find.

Library Functions

Library functions are ready-made functions provided with C++ in header files. Common examples are sqrt() and pow() from cmath, which find a square root and a power; strlen() from cstring, which finds the length of a string; and toupper() and tolower() from cctype, which change the case of a character. Using library functions saves time and avoids rewriting tested code.

Recursion

Recursion is the technique of a function calling itself to solve a smaller part of the same problem. A recursive function must have a base case that stops the recursion, otherwise it calls itself forever. The classic example is the factorial: factorial(n) = n * factorial(n - 1), with the base case factorial(0) = 1.

Code Example — Larger of Two Numbers

```
#include <iostream>
using namespace std;

// function definition: returns the larger of two numbers
int larger(int a, int b)
{
    if (a > b)
        return a;
    else
        return b;
}

int main()
{
    int x, y;
    cout << "Enter two numbers: ";
    cin >> x >> y;
    // function call: the values x and y are the arguments
    cout << "Larger = " << larger(x, y);
    return 0;
}
```

Unit 5: Functions — Practice Questions

Multiple Choice Questions

1. A function is

(a) a named block of code that performs one task (b) a loop (c) a keyword (d) a data type

2. The declaration of a function is also called its

(a) definition (b) prototype (c) call (d) body

3. The values passed to a function in a call are called

(a) returns (b) parameters (c) arguments (d) variables

4. A function that returns nothing has the return type

(a) char (b) int (c) float (d) void

5. A local variable is declared

(a) inside a function or block (b) in a header file (c) in the main menu (d) outside all functions

6. A global variable can be used by

(a) only one function (b) every function in the program (c) only the main function (d) no function

7. The library function sqrt() is defined in the header file

(a) iomanip (b) iostream (c) cmath (d) string

8. Recursion is a function that

(a) returns nothing (b) has no parameters (c) calls another function (d) calls itself

9. Every C++ program must have the function

(a) main() (b) run() (c) add() (d) start()

10. In pass by value, the function receives

(a) the address of the variable (b) a copy of the value (c) the name of the variable (d) nothing

Short Questions

1. Define a function.

Answer: A function is a named block of code that performs one task; it is written once and called whenever it is needed, which makes a program shorter, easier to read and easier to test.

2. What is a function prototype?

Answer: A function prototype is the declaration of a function before it is used; it gives the return type, the name and the types of the parameters, for example int add(int, int);, and it ends with a semicolon.

3. What is the difference between formal and actual parameters?

Answer: Formal parameters are the variables listed in the function definition, which receive values when the function is called, while actual parameters, or arguments, are the values passed in the function call, for example in add(5, 7) the values 5 and 7 are the actual parameters.

4. What is a local variable?

Answer: A local variable is declared inside a function or a block and can be used only there; it is created when the function runs and destroyed when the function ends.

5. What is recursion?

Answer: Recursion is the technique of a function calling itself to solve a smaller part of the same problem; it must have a base case that stops the recursion, as in factorial(n) = n * factorial(n - 1) with factorial(0) = 1.

Long Questions

1. Explain functions in C++ with the help of a program that finds the larger of two numbers.

Answer: A function is a named block of code that performs one task. It is declared with a prototype such as int larger(int, int);, defined with a body such as int larger(int a, int b) { if (a > b) return a; else return b; }, and used by calling it, for example larger(x, y). In the program, main() reads two numbers x and y, calls larger(x, y), and prints the result. The values x and y are the arguments, and a and b are the formal parameters that receive copies of them. Functions make the program shorter and easier to test, because the comparison is written once and can be reused.

2. Explain local and global variables with examples.

Answer: A local variable is declared inside a function or a block and can be used only there. For example, int sum = 0; declared inside main() is local to main(); it is created when main() runs and destroyed when it ends, and no other function can use it. A global variable is declared outside all functions, for example int taxRate = 16; written before main(), and it can be used by every function in the program for the whole run. Local variables are preferred, because a global variable can be changed by any function, which makes errors hard to find.

Unit 6: Digital Logic and Design

The Binary Number System

Computers work with the binary number system, which has only two digits, 0 and 1, called bits. A group of 8 bits is called a byte. A decimal number is converted to binary by repeatedly dividing by 2 and writing the remainders in reverse order; for example, 13 in decimal is 1101 in binary, because 13 = 8 + 4 + 0 + 1. Binary numbers are the language of the circuits inside a computer.

Logic Gates

A logic gate is a small electronic circuit that takes one or more binary inputs and produces one binary output according to a fixed rule. Gates are the building blocks of all digital circuits: processors, memory and every other part of a computer are made by joining gates together. The basic gates are AND, OR and NOT; the others are made by combining these.

The AND Gate

The AND gate gives an output of 1 only when ALL of its inputs are 1; otherwise the output is 0. Its symbol is a D-shaped figure, and it is written as A . B or A AND B. For two inputs A and B, the output is 1 only for the input pair 1, 1.

The OR Gate

The OR gate gives an output of 1 when ANY of its inputs is 1; the output is 0 only when all inputs are 0. Its symbol is a curved shape, and it is written as A + B or A OR B. For two inputs, the output is 0 only for the input pair 0, 0.

The NOT Gate

The NOT gate has only one input and one output. It inverts its input: 1 becomes 0 and 0 becomes 1. It is written as A with a bar over it, or as NOT A. Its symbol is a triangle with a small circle at the output, and the circle means inversion.

NAND, NOR and XOR Gates

The NAND gate is an AND gate followed by a NOT gate: its output is 0 only when all inputs are 1. The NOR gate is an OR gate followed by a NOT gate: its output is 1 only when all inputs are 0. The XOR (exclusive OR) gate gives 1 when its inputs are different and 0 when they are the same. NAND and NOR are called universal gates, because any other gate can be built from NAND gates alone or from NOR gates alone.

Boolean Algebra

Boolean algebra is the algebra of binary values, named after George Boole. Its basic laws are: the identity laws, A + 0 = A and A . 1 = A; the null laws, A + 1 = 1 and A . 0 = 0; the idempotent laws, A + A = A and A . A = A; the complement laws, A + A' = 1 and A . A' = 0; the commutative laws, A + B = B + A and A . B = B . A; the associative laws; the distributive laws; and De Morgan's laws, (A + B)' = A' . B' and (A . B)' = A' + B'. These laws are used to simplify digital circuits.

Half Adder and Full Adder

An adder is a circuit that adds binary numbers. A half adder adds two single bits and gives two outputs: the Sum, which is A XOR B, and the Carry, which is A AND B. A full adder adds three bits — A, B and a carry-in from the previous column — and gives a Sum and a Carry-out. Full adders are joined in a chain to add numbers of many bits, which is how the processor's arithmetic unit works.

Example — Truth Tables of the Basic Gates

```
AND gate:          OR gate:           NOT gate:
A B | Output       A B | Output       A | Output
0 0 | 0            0 0 | 0            0 | 1
0 1 | 0            0 1 | 1            1 | 0
1 0 | 0            1 0 | 1
1 1 | 1            1 1 | 1

XOR gate:
A B | Output
0 0 | 0
0 1 | 1
1 0 | 1
1 1 | 0
```

Unit 6: Digital Logic and Design — Practice Questions

Multiple Choice Questions

1. The binary number system uses the digits

(a) 0 to 7 (b) A to F (c) 0 and 1 (d) 0 to 9

2. One byte is equal to

(a) 16 bits (b) 32 bits (c) 4 bits (d) 8 bits

3. The AND gate gives 1 only when

(a) all inputs are 1 (b) all inputs are 0 (c) inputs are different (d) any input is 1

4. The OR gate gives 0 only when

(a) all inputs are 1 (b) all inputs are 0 (c) inputs are different (d) any input is 1

5. The NOT gate

(a) has two outputs (b) has two inputs (c) inverts its input (d) adds two bits

6. The XOR gate gives 1 when the inputs are

(a) both 0 (b) both 1 (c) the same (d) different

7. The NAND gate is an AND gate followed by a

(a) NOT gate (b) XOR gate (c) half adder (d) OR gate

8. Which gates are called universal gates?

(a) AND and OR (b) NAND and NOR (c) XOR and XNOR (d) NOT and AND

9. In a half adder, the Sum is

(a) A AND B (b) A OR B (c) A XOR B (d) NOT A

10. A full adder adds how many bits?

(a) four (b) one (c) two (d) three

Short Questions

1. Define a logic gate.

Answer: A logic gate is a small electronic circuit that takes one or more binary inputs and produces one binary output according to a fixed rule; gates are the building blocks of all digital circuits in a computer.

2. Write the truth table of the AND gate.

Answer: For inputs A and B, the AND gate gives: 0, 0 gives 0; 0, 1 gives 0; 1, 0 gives 0; 1, 1 gives 1. The output is 1 only when both inputs are 1.

3. What is a bit?

Answer: A bit is a single binary digit, 0 or 1; it is the smallest unit of data in a computer, and a group of 8 bits is called a byte.

4. What is the difference between a half adder and a full adder?

Answer: A half adder adds two single bits and gives a Sum (A XOR B) and a Carry (A AND B), while a full adder adds three bits — A, B and a carry-in — and gives a Sum and a Carry-out.

5. What is the NOT gate?

Answer: The NOT gate has one input and one output and inverts its input: 1 becomes 0 and 0 becomes 1. It is written as A with a bar over it, and its symbol is a triangle with a small circle at the output.

Long Questions

1. Explain the basic logic gates with their truth tables.

Answer: The basic gates are AND, OR and NOT. The AND gate gives 1 only when all inputs are 1: for inputs 0,0 the output is 0; 0,1 gives 0; 1,0 gives 0; 1,1 gives 1. The OR gate gives 1 when any input is 1: 0,0 gives 0; 0,1 gives 1; 1,0 gives 1; 1,1 gives 1. The NOT gate has one input and inverts it: 0 gives 1 and 1 gives 0. Other gates are made from these: NAND is AND followed by NOT, NOR is OR followed by NOT, and XOR gives 1 when the inputs are different. Gates are the building blocks of all digital circuits.

2. Explain the half adder and the full adder with truth tables.

Answer: A half adder adds two single bits, A and B, and gives two outputs: Sum = A XOR B and Carry = A AND B. Its truth table is: 0,0 gives Sum 0, Carry 0; 0,1 gives Sum 1, Carry 0; 1,0 gives Sum 1, Carry 0; 1,1 gives Sum 0, Carry 1. A half adder cannot take a carry from a previous column, so a full adder is used: it adds three bits — A, B and a carry-in — and gives a Sum and a Carry-out. Full adders are joined in a chain to add numbers of many bits, which is how the processor adds numbers.

Unit 7: Introduction to Scratch

What Is Scratch?

Scratch is a visual, block-based programming language developed by the MIT Media Lab for young learners. Instead of typing code, the programmer snaps together coloured blocks like pieces of a puzzle to make animations, stories and games. Scratch was launched in 2007 and is used in schools around the world, because it teaches the ideas of programming — sequence, selection, repetition and variables — without the need to remember exact spelling and punctuation.

The Scratch Interface

The Scratch editor has four main parts. The stage is the area where the program runs and the sprites act. The sprite list shows the characters and objects in the project. The blocks palette holds the coloured blocks, grouped by category. The script area is where blocks are joined together to make scripts. There are also tabs for costumes, which change how a sprite looks, and sounds, which add music and effects.

Sprites and the Stage

A sprite is a character or object in a Scratch project, such as a cat, a ball or a car. Each sprite has its own scripts, costumes and sounds. The stage is the background on which the sprites move and act; it has its own scripts and backdrops. A sprite moves on the stage using x and y coordinates: x runs left to right and y runs bottom to top, with the centre of the stage at (0, 0).

Blocks

- Motion: move, turn, glide, go to a position, change x or y.
- Looks: say, show, hide, change costume, change size, change effect.
- Sound: play a sound, change volume, set tempo.
- Events: when the green flag is clicked, when a key is pressed, when a sprite is clicked.
- Control: wait, repeat, forever, if-then, if-then-else, stop.
- Sensing: touching, asking a question, mouse position, timer.
- Operators: arithmetic, comparison and logical operations, and joining text.
- Variables: make a variable, set it, change it, show it on the stage.

Events — Starting a Script

A script in Scratch starts with an event block, called a hat block because of its rounded top. The most common is 'when green flag clicked', which runs the script when the green flag above the stage is pressed. Other event blocks start a script when a key is pressed, when a sprite is clicked, or when the project receives a broadcast message.

Loops in Scratch

Scratch has three loop blocks. 'repeat (10)' runs the blocks inside it a fixed number of times. 'forever' runs the blocks inside it again and again until the project is stopped, and is used for continuous movement and animation. 'repeat until' runs the blocks until its condition becomes true. Loops are placed around the blocks to be repeated, which makes the repetition easy to see.

Making an Animation

An animation is made by changing a sprite's costume, position or size over time. For example, a walking cat is made by switching between two costumes inside a forever loop, with a short wait between the switches, while the sprite moves a few steps each time. A 'glide' block moves a sprite smoothly from one place to another in a given time.

Making a Simple Game

A simple game is made by combining events, loops, sensing and variables. For example, a sprite moves with the arrow keys using 'when key pressed' blocks; a second sprite moves on its own inside a forever loop; the 'if touching' block checks a collision, and a variable called Score is increased by 1 each time the moving sprite touches a star. The game ends when the score reaches a target, or when a timer runs out.

Example — A Sprite That Moves and Bounces

```
when green flag clicked
forever
    move (10) steps
    if on edge, bounce
    wait (0.1) seconds

This script makes the sprite move across the stage
forever, turn back when it reaches an edge, and
pause briefly between steps so the movement is smooth.
```

Unit 7: Introduction to Scratch — Practice Questions

Multiple Choice Questions

1. Scratch was developed by

(a) MIT Media Lab (b) Apple (c) Microsoft (d) Google

2. Scratch is a

(a) text-based language (b) block-based visual language (c) machine language (d) database language

3. A sprite in Scratch is a

(a) programming error (b) background (c) character or object in a project (d) sound effect

4. The stage in Scratch is the

(a) sound library (b) script area (c) list of blocks (d) area where the program runs

5. The script that runs when the green flag is clicked starts with the block

(a) when green flag clicked (b) forever (c) move 10 steps (d) repeat 10

6. The forever block

(a) runs a fixed number of times (b) runs again and again until stopped (c) stops the project (d) runs once

7. The motion category of blocks is used to

(a) ask a question (b) play sounds (c) move a sprite (d) change a costume

8. A costume in Scratch is

(a) a variable (b) a stage background (c) a sound (d) how a sprite looks

9. A variable in Scratch is used to

(a) store a value such as a score (b) move a sprite (c) play a sound (d) change a backdrop

10. The repeat block is used to

(a) change a costume (b) run blocks a fixed number of times (c) stop the project (d) ask a question

Short Questions

1. What is Scratch?

Answer: Scratch is a visual, block-based programming language developed by the MIT Media Lab, in which the programmer snaps together coloured blocks to make animations, stories and games instead of typing code.

2. What is a sprite?

Answer: A sprite is a character or object in a Scratch project, such as a cat, a ball or a car; each sprite has its own scripts, costumes and sounds, and it moves and acts on the stage.

3. What is the stage?

Answer: The stage is the area where a Scratch program runs and the sprites act; it has its own scripts and backdrops, and a sprite moves on it using x and y coordinates, with the centre at (0, 0).

4. What does 'when green flag clicked' do?

Answer: 'When green flag clicked' is an event block that starts the script joined below it when the green flag above the stage is pressed; it is the most common way to start a Scratch program.

5. Name four block categories in Scratch.

Answer: Four block categories in Scratch are Motion, Looks, Sound and Control; the others are Events, Sensing, Operators, Variables and My Blocks.

Long Questions

1. Describe the Scratch interface and its main parts.

Answer: The Scratch editor has four main parts. The stage is the area where the program runs and the sprites act, with its own backdrops. The sprite list shows the characters and objects in the project, and each sprite has its own scripts, costumes and sounds. The blocks palette holds the coloured blocks grouped by category: Motion, Looks, Sound, Events, Control, Sensing, Operators and Variables. The script area is where blocks are joined together to make scripts. There are also tabs for costumes, which change how a sprite looks, and sounds, which add music and effects.

2. Explain how to make a simple animation in Scratch.

Answer: To make a simple animation, first choose or draw a sprite and a backdrop on the stage. Then join blocks in the script area: start with 'when green flag clicked', then use a forever loop so the animation continues. Inside the loop, move the sprite a few steps with 'move (10) steps', add 'if on edge, bounce' so it turns back at the edge, and add 'wait (0.1) seconds' so the movement is smooth. To make the sprite look as if it is walking, add 'next costume' inside the loop so its costume changes each time. Pressing the green flag runs the animation, and the red stop sign ends it.

MCQ Answer Key

Each unit has 10 MCQs, numbered 1 to 10.

Unit	1	2	3	4	5	6	7	8	9	10
Unit 1	A	B	C	D	A	B	C	D	A	B
Unit 2	C	D	A	B	C	D	A	B	C	D
Unit 3	A	B	C	D	A	B	C	D	A	B
Unit 4	C	D	A	B	C	D	A	B	C	D
Unit 5	A	B	C	D	A	B	C	D	A	B
Unit 6	C	D	A	B	C	D	A	B	C	D
Unit 7	A	B	C	D	A	B	C	D	A	B

Score guide: count your correct answers out of 10 for each unit. Re-read the notes for every unit in which you scored less than 7 out of 10.

End of the book. Good luck with your studies — revise regularly and check bsek.edu.pk for the current model paper.