PYQ-PSPC

What is subscripted variable?

A subscripted variable is a variable that has a subscript (index number) to access a specific value from a collection of values stored under one name.

How do you use a decision table? Explain.

A decision table is a tool used to represent different conditions and the actions that should be taken for each condition.

Conditions	Rule 1	Rule 2	Rule 3
Condition 1	Yes	Yes	No
Condition 2	Yes	No	Yes
Action	A	B	C

What is meant by coupling modules? Explain.

Coupling tells us how strongly one module depends on another module.

Suppose we have two modules:

Module A → Calculates total marks

Module B → Prints report

What is cohesion? Explain the different types of cohesion.

Cohesion tells us how closely related the functions within a single module are.

High cohesion → Good design

Low cohesion → Poor design

Example of High Cohesion (Good Design):

Module: CalculateSalary

This module only performs salary-related tasks:

- Calculate basic salary
- Calculate bonus
- Calculate tax
- Return final salary

All functions are related to **one purpose → salary calculation**.

Example of Low Cohesion (Bad Design)

Module: CalculateSalary

This module performs:

- Calculate marks
- Print student report
- Connect to database
- Send email

different types of cohesion

- **Coincidental Cohesion**

Unrelated tasks are grouped together in one module.

No meaningful connection between functions.

Example:

A module that prints a report, calculates salary, and updates attendance — all in one.

- **Logical Cohesion**

Similar types of operations are grouped together.

Selection is controlled by a flag.

Example: function process(int option)

If option = 1 → Print

If option = 2 → Save

- **Temporal Cohesion**

Tasks that are related by timing are grouped together.

Example

Initialization tasks at program start.

Closing files at program end.

- **Procedural Cohesion**

Elements are related because they follow a sequence of steps.

Example

Read data → Process data → Print data

- **Communicational Cohesion**

Functions operate on the same data.

Example

A module that reads student data and updates the same student record.

- **Sequential Cohesion**

Output of one part becomes input of another part.

Read marks → Calculate total → Calculate percentage

- **Functional Cohesion (Best)**

All parts of the module work together to perform a single, well-defined task.

Example: A module only for calculating total marks.

Differentiate between primary key and secondary key.

primary key is a field (or combination of fields) that uniquely identifies each record in a table.

Features:

Must be unique

Cannot be NULL

Only one primary key per table

Secondary Key

A secondary key is a field used to search or retrieve data, but it may not be unique.

Features:

Can have duplicate values

Can be NULL

A table can have multiple secondary keys

Explain call by value and call by reference with suitable example.

In call by value, a copy of the actual value is passed to the function.

- Changes made inside the function do not affect the original variable.
- Original data remains unchanged.

Example:

```
#include <stdio.h>
void change(int x) {
   x = 20;
}
int main() {
   int a = 10;
   change(a);
   printf("%d", a);
   return 0;
}
```

In call by reference, the address of the variable is passed to the function.

- Changes made inside the function affect the original variable.
- Original value gets modified.

Example:

```
#include <stdio.h>
void change(int *x) {
    *x = 20;
}
int main() {
    int a = 10;
    change(&a);
    printf("%d", a);
    return 0;
}
```

Write a program to swap the two number by using third variable with both call by value an reference.

```
#include <stdio.h>
void swap(int a, int b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
    printf("\nInside function (Call by Value): a = %d, b = %d", a, b);
}
int main() {
    int x = 10, y = 20;
    printf("Before swap: x = %d, y = %d", x, y);
    swap(x, y);
    printf("\nAfter swap (in main): x = %d, y = %d", x, y);
    return 0;
}
```

```c
#include <stdio.h>

void swap(int *a, int *b) {
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}

int main() {
    int x = 10, y = 20;

    printf("Before swap: x = %d, y = %d", x, y);
    swap(&x, &y);
    printf("\nAfter swap (Call by Reference): x = %d, y = %d", x, y);

    return 0;
}
```

Difference between functions and macros.

Macro

A macro is defined using #define and is handled by the preprocessor before compilation.

Example:

```
#include <stdio.h>
#define ADD(a, b) (a + b)
int main() {
    int x = 10, y = 20;
    int result = ADD(x, y);
    printf("Sum = %d", result);
    return 0;
}
```

- **Function**

A function is a block of code that performs a specific task and is executed when it is called.

```
#include <stdio.h>
int add(int a, int b) {
   return a + b;
}
int main() {
   int sum;
// Function call
   sum = add(5, 6);
// Display result
   printf("Sum = %d", sum);
   return 0;
}
```

What is Pseudocode?

Pseudocode is a step-by-step description of an algorithm written in simple English without programming syntax.

Example: Add Two Numbers

```
START
READ number1
READ number2
sum = number1 + number2
PRINT sum
END
```

Write a 'C' program to read N numbers into an array and perform linear search.

```
#include <stdio.h>
int main() {
    int n, i, key, found = 0;
   printf("Enter number of elements: ");
    scanf("%d", &n);
    int arr[n];
  printf("Enter %d numbers:\n", n);
    for(i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
printf("Enter number to search: ");
    scanf("%d", &key);
for(i = 0; i < n; i++) {
        if(arr[i] == key) {
            printf("Element found at position %d\n", i + 1);
            found = 1;
            break;
        }
    }
    if(found == 0) {
        printf("Element not found\n");
    }
    return 0;
}
```

Write a 'C' program to demonstrate the use of unconditional goto statement.

#include <stdio.h>

```
int main()
 {
    int num;
      start:
      printf("Enter a positive number: ");
      scanf("%d", &num);
    if(num < 0)
      {
        printf("You entered a negative number. Try again.\n");
        goto start;
    }
    printf("You entered: %d", num);
    return 0;
}
```

Differentiate between typecasting and type conversion.

Type conversion is done automatically by the compiler.

- Also called **implicit conversion**
- No need to write extra code
- Usually converts smaller data type to larger data type

Example:

```
#include <stdio.h>
int main() {
    int a = 10;
    float b;
    b = a;   // Automatic conversion (int → float)
    printf("Value of b = %f", b);
    return 0;
}
```

Typecasting is done manually by the programmer.

- Also called **explicit conversion**
- Programmer forces conversion
- Syntax: (datatype) variable

Example:

```
#include <stdio.h>
int main() {
   float a = 10.75;
   int b;
   b = (int)a;   // Manual conversion
   printf("Value of b = %d", b);
   return 0;
}
```