ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

C Programming Lab (1BPOPL107/207)

2025-26

Prepared By,

Deepashri K S
Asst. Professor
IS&E Department

Approved by

Dr. Sampath S
Prof & Head
IS&E Department

Lab Program List

	PART-A	
Sl. No.	**CONVENTIONAL EXPERIMENTS**	**Page No.**
1	A robot needs to find how far it must travel between two points on a 2D plane. Develop a C program to calculate the straight-line distance between the given coordinates.	
2	Develop a C program that takes a student's marks as input and displays their grade based on the following criteria: 90 and above: Grade A 75 to 89: Grade B 60 to 74: Grade C 50 to 59: Grade D Below 50: Grade F Choose a suitable control structure to implement this logic efficiently.	
3	Develop a C program that takes a unique identification input like PAN Number, AADHAR_Number, APAAR_Id, Driving License, and Passport and checks it against a set of stored KYC records. Based on the input, display whether the individual is verified or not. Use an appropriate control structure to handle multiple possible ID matches. Assume all Unique identification are of integer type.	
4	A math app needs to determine the type of roots for a quadratic equation based on user input. Develop a C program to calculate and display the roots based on the given coefficients	
5	A sensor in a robotic arm needs to calculate the angle of rotation in real-time, but the hardware doesn't support built-in trigonometric functions. Develop a C program to approximate the value of sin(x) using a series expansion method for improved performance.	
6	Develop a C program that accepts a course description string and a keyword from the user. Search whether the keyword exists within the course description using appropriate string functions. If found, display: "Keyword " found in the course description." Otherwise, display: "Keyword " not found in the course description."	
7	Develop a C program that takes marks for three subjects as input. Use a function to check if the student has passed (minimum 40 marks in each subject). Display the average and whether the student passed or failed.	
8	In an ATM system, two account balances need to be swapped temporarily for validation. Develop a C program that accepts two balances and uses a function with pointers to swap them. Display the balances before and after swapping.	

PART-B		
S.No	**TYPICAL OPEN-ENDED EXPERIMENTS**	**Page No.**
1	A college library has a digital bookshelf system where each book is assigned a unique Book ID. The bookshelf is organized in ascending order of Book IDs. Develop a C Program to quickly find whether a book with a specific Book ID is available in the shelf.	
2	A sports teacher has recorded the scores of students in a 100-meter race. To prepare the result sheet, the teacher wants the scores arranged in descending order (from highest to lowest). Develop a C program to sort the scores.	
3	A small warehouse tracks how many units of different products are shipped from multiple branches. Another dataset shows how much revenue each product generates per unit. Develop a C program which combines these datasets to calculate the total revenue generated by each branch.	
4	A basic mobile contact manager stores first and last names separately. For displaying full names in the contact list, you need to join them manually. Additionally, the system must check the length of each full name to ensure it fits the screen. Perform these operations by developing a C program without using built in string functions.	
5	A currency exchange booth allows users to convert between two currencies. Before confirming the exchange, the system simulates a swap of the values to preview the result without actually changing the original data. In other cases, it updates the actual values. Develop a C program that implements both behaviours using Call by Value and Call by reference	
6	A local library needs to store and display details of its books, including title, author, and year of publication. Design a structure that can hold these details and develop a C program to display a list of all books entered.	

ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Institution Vision and Mission

Vision

To develop Adichunchanagiri Institute of Technology as a center of excellence and to strive for continuous improvement of technical education and human resource advancement.

Mission

To achieve Excellence in Education, Entrepreneurship, and Innovation by producing Engineers with high Ethical Standards, Integrity, and Credibility.

ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Department Vision and Mission

Vision

To be recognized as a center of excellence in information technology and allied areas with quality learning and research environment

Mission

- Provide intellectual & professional leadership in ethical and social areas pertaining to information in contemporary society.
- Advancing the state of knowledge of information studies through research and development.
- Providing a platform to discuss cutting edge technologies.

ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Subject: C Programming Lab
Subject code: 1BPOPL107/207
Semester: I Year

Course Outcomes

At the end of the course the student will be able to:

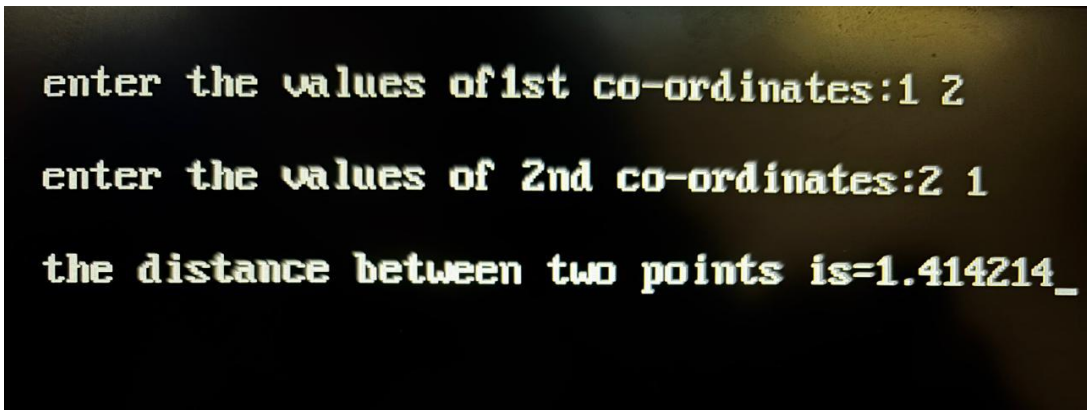
CO1	Develop programs in C to solve simple computational problems.
CO2	Make use of C language derived data types to solve simple real-world problems.
CO3	Build a document consisting of experiment setup, design, implementation and results with

PART-A

1. **A robot needs to find how far it must travel between two points on a 2D plane. Develop a C program to calculate the straight-line distance between the points**

```
#include <stdio.h>
#include <math.h>
void main()
 {
    float x1, y1, x2, y2, distance;
    printf("Enter x1 and y1 (coordinates of the first point): ");
    scanf("%f %f", &x1, &y1);
    printf("Enter x2 and y2 (coordinates of the second point): ");
    scanf("%f %f", &x2, &y2);
    distance = sqrt((x2 - x1)*(x2 - x1) + (y2 - y1)*(y2 - y1));
    printf("The straight-line distance between the two points is: %.2f\n", distance);
}
```

Output:

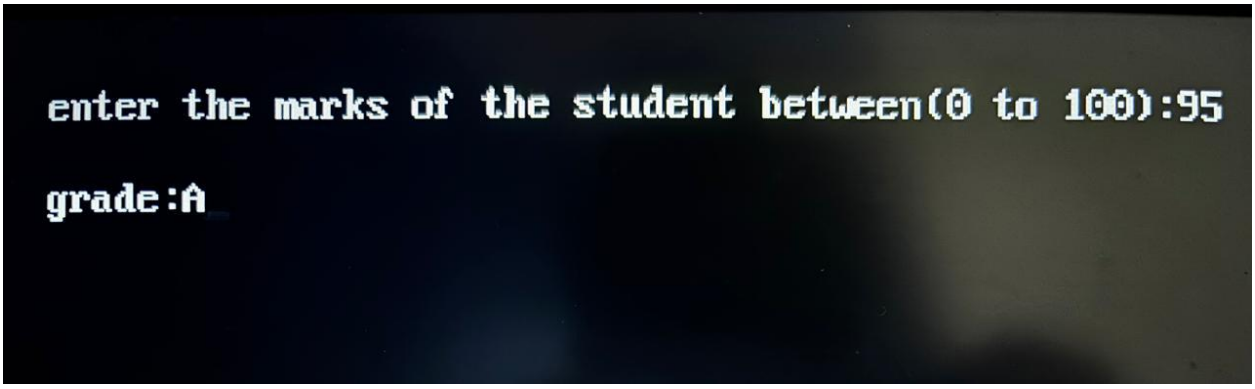


2. **Develop a C program that takes a student's marks as input and displays their grade based on the following criteria: 90 and above: Grade A 75 to 89: Grade B 60 to 74: Grade C 50 to 59: Grade D Below 50: Grade F Choose a suitable control structure to implement this logic efficiently.**

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int  marks;
    clrscr();
    printf("\n Enter the student marks(0 to 100)");
    scanf("%d",&marks);
    if(marks<0||marks>100)
        printf("\n Invalid marks!pelase enter a value between 0 to 100");
    else if(marks>=90)
          printf("\nGrade:A);
    else if(marks>=75)
          printf("\nGrade:B);
    else if(marks>=60)
          printf("\nGrade:C);
    else if(marks>=50)
          printf("\nGrade:D);
    else
           printf("\nGrade:F);
    getch();
    return 0;
```

```
}
```

Output:

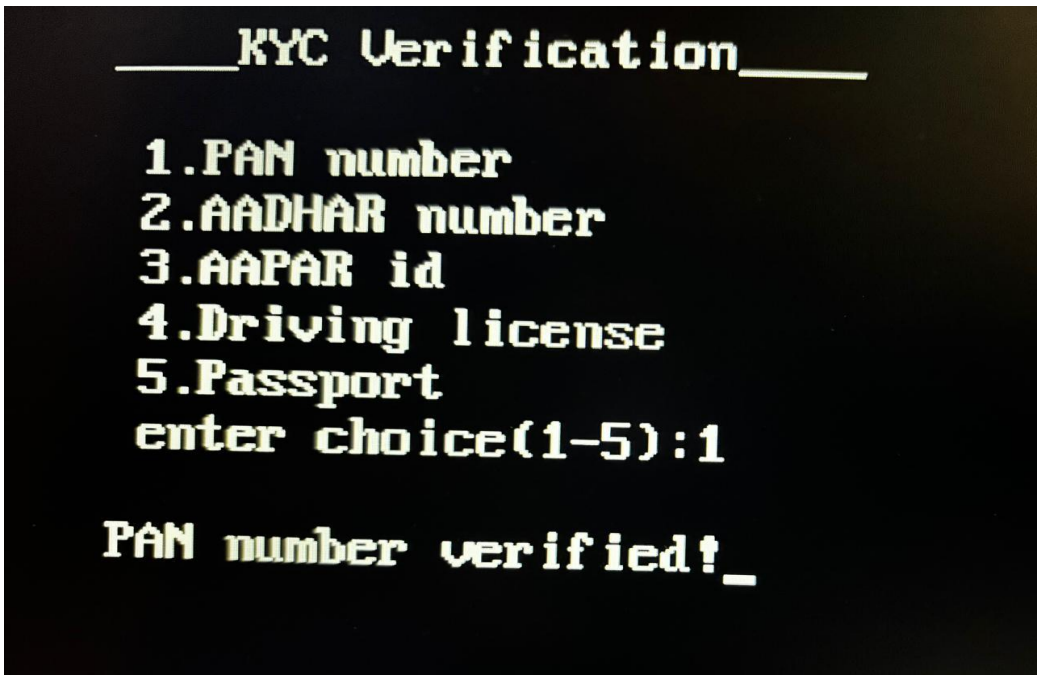


3. Develop a C program that takes a unique identification input like PAN Number, AADHAR_Number, APAAR_Id, Driving License, Passport and checks it against a set of stored KYC records. Based on the input, display whether the individual is verified or not. Use an appropriate control structure to handle multiple possible ID matches. Assume all Unique identification are of integer type.

```
#include <stdio.h> int main()

{
   int choice, id;
   printf("------ KYC Verification System --- \n");
   printf("1. PAN Number\n"); printf("2. AADHAR Number\n"); printf("3. APAAR
   Id\n"); printf("4. Driving License\n"); printf("5. Passport\n");
   printf("Enter your choice (1-5): "); scanf("%d", &choice); switch(choice)
   {
      case 1:   // PAN
            printf("PAN Verified!\n"); break;
     case 2: // AADHAAR
               printf("AADHAR Verified!\n");
              break;
     case 3: // APAAR
              printf("APAAR Verified!\n");
              break;
    case 4: // Driving License
               printf("Driving License Verified!\n");
               break;
    case 5: // Passport
```

```
            printf("Passport Verified!\n");
            break;
    default:  printf("Not Verified!\n");
  }
  return 0;
}
```

Output:

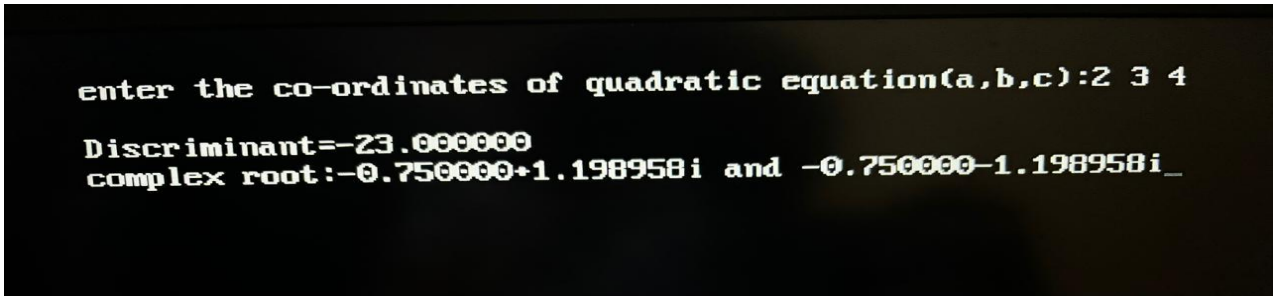


4. A math app needs to determine the type of roots for a quadratic equation based on user input. Develop a C program to calculate and display the roots based on the given coefficients.

```
#include        <stdio.h>
#include  <math.h>  int
main()
{
      float   a, b, c, root1, root2, disc, real, imag;
      printf("Enter coefficients of quadratic equation (a, b, c): "); scanf("%f
      %f %f", &a, &b, &c);
      if (a == 0 && b == 0)
  {
     printf("Invalid Coefficients!\n");
  }
  else if (a == 0)
  {
  // Linear equation bx + c = 0
       root1 = -c / b;
       printf("Linear Equation with root: %.2f\n", root1);
   }
else
{
// Qudratic equation
     disc = b * b - 4 * a * c;
     printf("Discriminant = %.2f\n", disc);
    if (disc > 0)
    {
```

```
        root1 = (-b + sqrt(disc)) / (2 * a);
        root2 = (-b - sqrt(disc)) / (2 * a);
        printf("Two distinct real roots: %.2f and %.2f\n", root1, root2);
    }
    else if (disc == 0)
    {
        root1 = -b / (2 * a);
        printf("One real root: %.2f\n", root1);
    }
    else
    {
        real = -b / (2 * a);
        imag = sqrt(-disc) / (2 * a);
        printf("Complex roots: %.2f + %.2fi and %.2f - %.2fi\n", real, imag, real, imag);
    }
}
return 0;
}
```

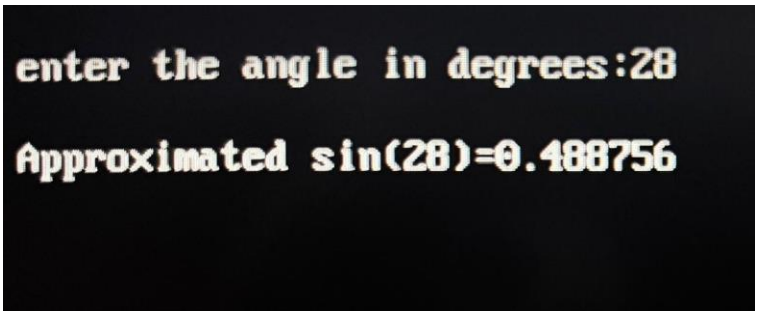
Output:



5. A sensor in a robotic arm needs to calculate the angle of rotation in real-time, but the hardware doesn't support built-in trigonometric functions. Develop a C program to approximate the value of sin(x) using a series expansion method for improved performance.

```
#include <stdio.h>

 #include<math.h>

#define PI  3.14

void main()

{

    float   sum,term,x,nume;

    int  deg;

    i=2,fact=1;

    printf("Enter angle in degrees: ");

    scanf("%d", &deg);

    x=(deg*PI)/180;

     sum=x;

      nume=x;

     do

      {

          fact=fact*i*(i+1);

          nume = -nume * x * x;

          term = nume/fact;

          sum += term;

           i+=2;

      } while (fabs(term)>=0.0001);

  printf("Approximated sin(%d) = %.2f\n", deg, sum);
}
```

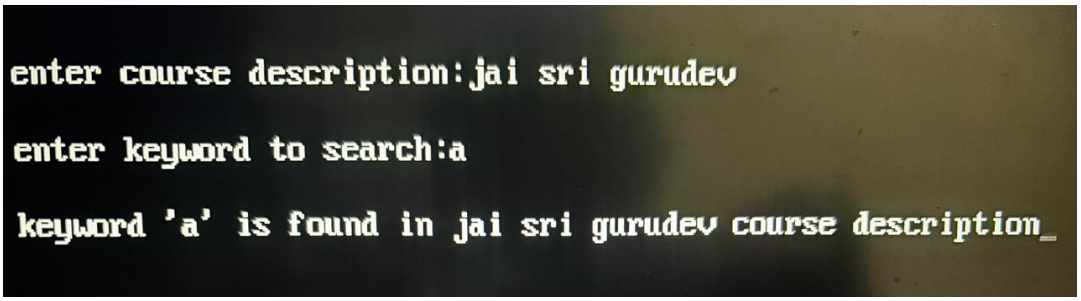
Output:



6. Develop a C program that accepts a course description string and a keyword from the user. Search whether the keyword exists within the course description using appropriate string functions. If found, display: "Keyword '' found in the course description." Otherwise, display: "Keyword '' not found in the course description.

```
#include <stdio.h>
#include <string.h>
#define MAX 100
int main()
 {
    char description[MAX];
    char keyword[100];
    int len;
    printf("Enter the course description:\n");
    scanf("%s",description);
    len = strlen(description);
    printf("Enter the keyword to search: ");
    scanf("%s", keyword);
    if (strstr(description, keyword) )
     printf("Keyword '%s' found in the %s course description.\n", keyword,description);
    else
        printf("Keyword '%s' not found in the %s course description.\n", keyword,description);
     return 0;
 }
```

Output:

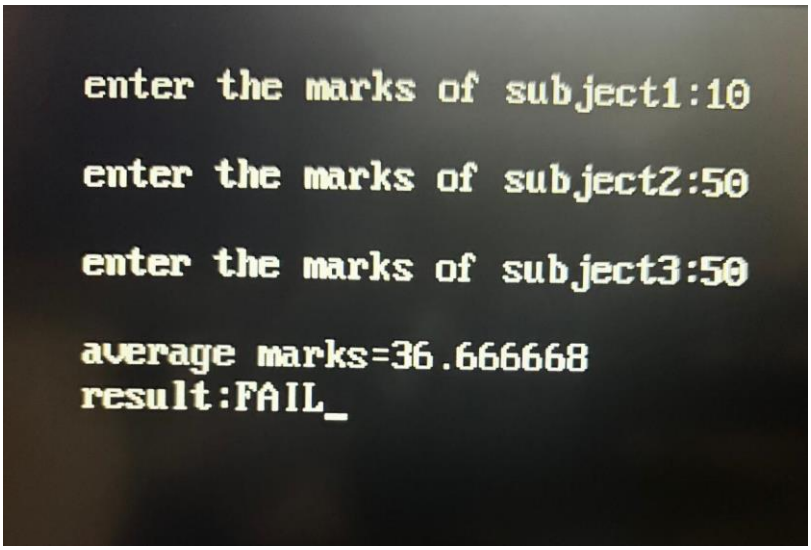

enter course description:jai sri gurudev
enter keyword to search:a
keyword 'a' is found in jai sri gurudev course description_

7. Develop a C program that takes marks for three subjects as input. Use a function to check if the student has passed (minimum 40 marks in each subject). Display the average and whether the student passed or failed.

```
#include <stdio.h>
int isPassed(int m1, int m2, int m3)
{
        if (m1 >= 40 && m2 >= 40 && m3 >= 40)
            return 1;
        else
            return 0;
}
void main()
{
        int  subject1, subject2, subject3; float
        average;
        printf("Enter marks for Subject 1: ");
        scanf("%d", &subject1);
        printf("Enter marks for Subject 2: ");
        scanf("%d", &subject2);
        printf("Enter marks for Subject 3: ");
        scanf("%d", &subject3);
        average = (subject1 + subject2 + subject3) / 3.0;
        printf("Average Marks: %.2f\n", average);
        if (isPassed(subject1, subject2, subject3))
            printf("Result: PASS\n");
```

```
        else
            printf("Result: FAIL\n");
}
```

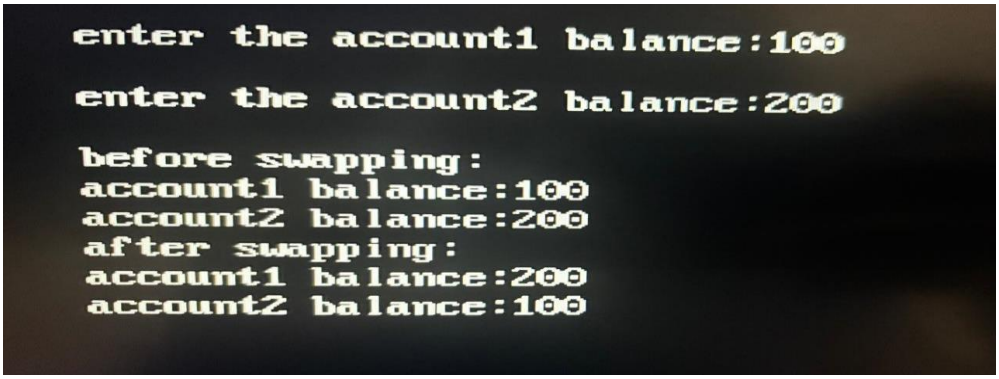
Output:



8. In an ATM system, two account balances need to be swapped temporarily for validation. Develop a C program that accepts two balances and uses a function with pointers to swap them. Display the balances before and after swapping.

```
#include <stdio.h>
void swap(int *a, int *b)
{
int  temp = *a;
*a = *b;
*b = temp;
}
void main()
{
int   balance1, balance2;
printf("Enter balance for Account 1: ");
scanf("%d", &balance1);
printf("Enter balance for Account 2: ");
scanf("%d", &balance2);
printf("\nBefore Swapping:\n");
printf("Account 1 Balance: %d\n", balance1);
 printf("Account 2 Balance: %d\n", balance2);
swap(&balance1, &balance2);
printf("\nAfter Swapping:\n");
 printf("Account 1 Balance: %d\n", balance1);
printf("Account 2 Balance: %d\n", balance2);
}
```

Output:


enter the account1 balance:100
enter the account2 balance:200
before swapping:
account1 balance:100
account2 balance:200
after swapping:
account1 balance:200
account2 balance:100

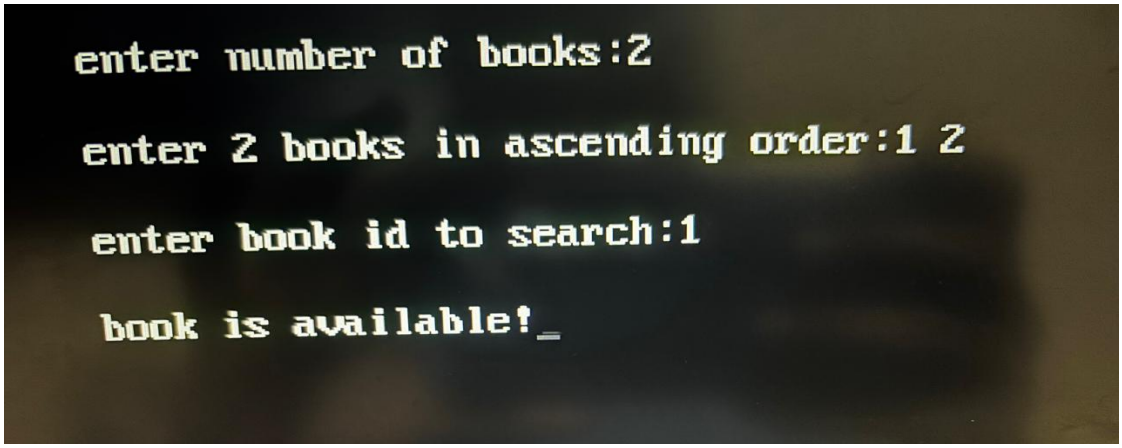
Part B

1. A college library has a digital bookshelf system where each book is assigned a unique Book ID. The bookshelf is organized in ascending order of Book IDs. The task is to develop a C program to quickly find whether a book with a specific Book ID is available in the shelf using binary search.

```
#include <stdio.h>

#include<conio.h>

int main()

{

    int   n, i, key, low, high, mid, found = 0;

    clrscr();

    printf("Enter number of books: ");

    scanf("%d", &n);

    int  books[n];

    printf("Enter %d Book IDs in ascending order:\n", n);

    for(i = 0; i < n; i++)

    scanf("%d", &books[i]);

    printf("Enter Book ID to search: ");

    scanf("%d", &key);

      low = 0;

    high = n - 1;

    while(low <= high)

    {

        mid = (low + high) / 2;
```

```
    if(books[mid] == key)
     {
          found = 1;
          break;
     }
    else if(books[mid] < key)
               low = mid + 1;
          else
               high = mid - 1;
     }
    if(found)
         printf("Book is available.\n");
    else
         printf("Book is not available.\n");
    getch();
    return 0;
}
```

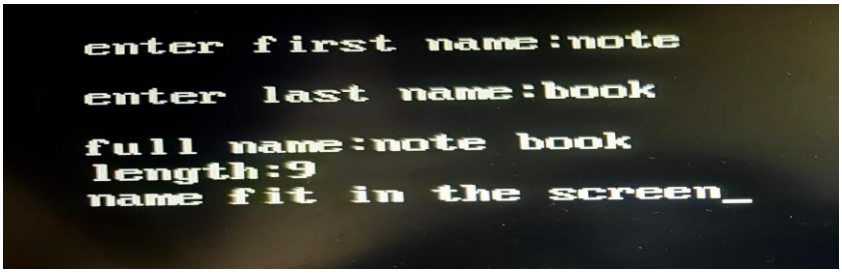
Output:



2. A basic mobile contact manager stores first and last names separately. For displaying the full names in the contact list, the system needs to join them manually. Additionally, the system must check the length of each full name to ensure it fits within the screen. The task is to implement these operations in C without using built-in string functions.

```
#include <stdio.h>
int  main()
{
    char  first[50], last[50], full[100];
    int   i = 0, j = 0, length = 0;
    printf("Enter first name: ");
    scanf("%s", first);
    printf("Enter last name: ");
    scanf("%s", last);
    while(first[i] != '\0')
    {
       full[length++] = first[i++];
    }

    full[length++] = ' ';
    while(last[j] != '\0')
    {
       full[length++] = last[j++];
     }
    full[length] = '\0';
    printf("Full Name: %s\n", full);
    printf("Length: %d\n", length);
    if(length > 20)
       printf("Name too long for screen.\n");
    else
       printf("Name fits the screen.\n");
    return 0;

}
```

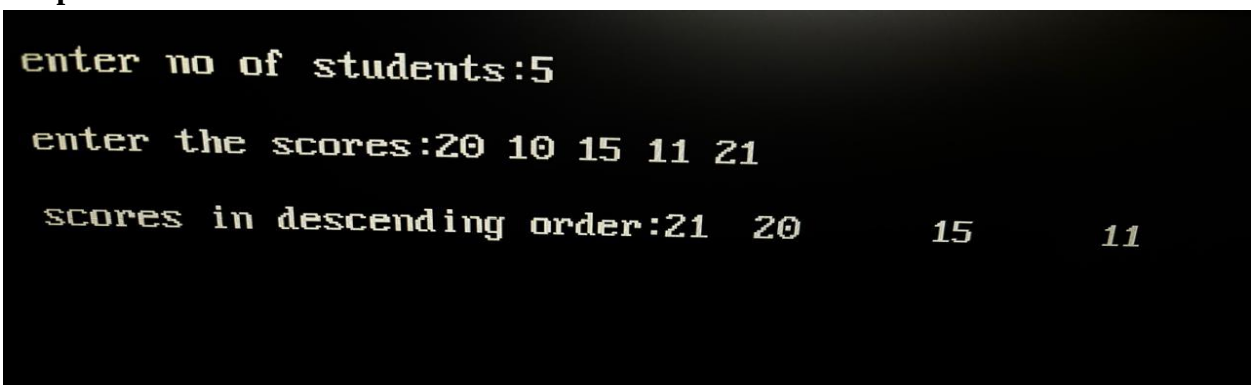
Output:



3. A sports teacher has recorded the scores of students in a 100-meter race. To prepare the result sheet, the teacher wants the scores arranged in descending order (from highest to lowest). The task is to develop a C program to sort the scores using Bubble Sort.

```
#include <stdio.h>
int main()
{
    int  n, i, j, temp;
    printf("Enter number of students: ");
    scanf("%d", &n);
    int scores[n];
    printf("Enter the scores:\n");
    for(i = 0; i < n; i++)
    scanf("%d", &scores[i]);
    for(i = 0; i < n-1; i++)
    {
      for(j = 0; j < n-i-1; j++)
      {
        if(scores[j] < scores[j+1])
        {
           temp = scores[j];
           scores[j] = scores[j+1];
           scores[j+1] = temp;
        }
      }
    }

     printf("Scores in descending order:\n");
    for(i = 0; i < n; i++)
    printf("%d \t", scores[i]);
    return 0;
}
```

Output:



4 Mobile Contact Manager (Concatenating Names) Problem Statement A basic mobile contact manager stores first and last names separately. For displaying the full names in the contact list, the system needs to join them manually. Additionally, the system must check the length of each full name to ensure it fits within the screen. The task is to implement these operations in C without using built-in string functions.

```
#include <stdio.h>
int main()
{
 char first[50], last[50], full[100];
 int i = 0, j = 0, length = 0;
 printf("Enter first name: ");
 scanf("%s", first);
 printf("Enter last name: ");
 scanf("%s", last);
 // Copy first name
 while(first[i] != '\0')
 {
 full[length++] = first[i++];
 }
 // Add space
 full[length++] = ' ';
// Add space
 full[length++] = ' ';
 // Copy last name
 while(last[j] != '\0')
 {
 full[length++] = last[j++];
```

```
}
 // Terminate string
 full[length] = '\0';
 // Output
 printf("Full Name: %s\n", full);
 printf("Length: %d\n", length);
 if(length > 20)
 printf("Name too long for screen.\n");
 else
 printf("Name fits the screen.\n");
 return 0;
}
```

Output:

```
Enter first name: Information
Enter last name: Science
Full Name: Information  Science
Length: 20
Name fits the screen.
```

5. A currency exchange booth allows users to convert between two currencies. Before confirming the exchange, the system simulates a swap of the values (preview) without actually changing the original data. In other cases, it updates the actual values (permanent swap). The task is to implement both behaviors.

```
#include <stdio.h>

void swapByValue(int a, int b)

{

 int temp = a;

a = b;

 b = temp;

 printf("Preview Swap (By Value): %d %d\n", a, b);

}

void swapByReference(int *a, int *b)

{

 int temp = *a;

 *a = *b;

 *b = temp;

 printf("Actual Swap (By Reference): %d %d\n", *a, *b);

}

int main()

{

 int x, y;

 printf("Enter two currency values: ");

 scanf("%d %d", &x, &y);

 swapByValue(x, y);

 printf("After Call by Value: %d %d\n", x, y);

 swapByReference(&x, &y);
```

```
 printf("After Call by Reference: %d %d\n", x, y);

 return 0;

}
```

Output:

```
Enter two currency values: 10 20
 Swap By Value: 20 10
After Call by Value: 10 20
Actual Swap (By Reference): 20 10
After Call by Reference: 20 10
```

6. A local library needs to store and display details of its books, including title, author, and year of publication. The task is to design a structure that can hold these details and develop a C program to display the list of books entered.

```
#include <stdio.h>
struct Book
{
 char title[50];
 char author[50];
 int year;
};
int main()
{
 int n, i;
 printf("Enter number of books: ");
 scanf("%d", &n);
 struct Book booklist[n];
 for(i = 0; i < n; i++)
 {
 printf("Enter title, author, and year for book %d:\n", i+1);
 scanf("%s %s %d", booklist[i].title, booklist[i].author, & booklist[i].year);
 }
 printf("\nLibrary Book Details:\n");
 for(i = 0; i < n; i++)
 {
 printf("Book %d: %s by %s (%d)\n", i+1, booklist[i].title, booklist[i].author,
booklist[i].year);
```

```
 }
 return 0;
}
```

Output:

```
Enter number of books: 1
Enter title, author, and year for book 1:
ProgrammingC  Balaguruswamy  1980

Library Book Details:
Book 1: ProgrammingC by Balaguruswamy (1980)
```

Viva Questions

1. What is C language?
C is a procedural programming language used to develop system software and application programs.

2. Who developed C language?
C was developed by **Dennis Ritchie** in 1972 at **Bell Labs**.

3. Why is C called a middle-level language?
Because it supports both low-level features (like pointers) and high-level features (like functions).

4. What is the structure of a C program?
Documentation section → Preprocessor directives → Global declaration → `main()` function → User-defined functions.

5. What is the use of #include<stdio.h>?
It includes the standard input-output header file for functions like `printf()` and `scanf()`.

6. What is main() function?
It is the starting point of execution of a C program.

7. What are data types in C?
Data types define the type of data a variable can store (int, float, char, double).

8. What is the difference between int and float?
`int` stores whole numbers.`float` stores decimal numbers.

9. What is a variable?
A variable is a named memory location used to store data.

10. What is a constant?
A constant is a value that cannot be changed during program execution.

11. What are operators in C?
Operators are symbols used to perform operations (like +, -, *, /).

12. What is the difference between = and == ?
= is assignment operator. == is comparison operator.

13. What are control statements?
They control the flow of execution (if, switch, for, while, do-while).

14. What is a loop?
A loop is used to repeat a block of code multiple times.

15. What is an array?
An array is a collection of similar data types stored in continuous memory locations.

16. What is a function?
A function is a block of code that performs a specific task.

17. What is a pointer?
A pointer is a variable that stores the address of another variable.

18. What is the difference between call by value and call by reference?
Call by value → copy of variable is passed.
Call by reference → address of variable is passed.

19. What is a structure in C?
A structure is a user-defined data type that groups different data types together.

20. What is the use of return 0; in main()?
It indicates that the program executed successfully.

21. What is a header file?
A header file contains function declarations and macros. Example: `stdio.h`.

22. What is a preprocessor?
The preprocessor processes directives like `#include` and `#define` before compilation.

23. What is #define?
It is used to define symbolic constants.
Example: `#define PI 3.14`

24. What is the difference between local and global variables?
Local variables are declared inside a function.
Global variables are declared outside all functions.

25. What is scope of a variable?
Scope defines where a variable can be accessed in a program.

26. What is lifetime of a variable?
Lifetime is the time duration during which a variable exists in memory.

27. What are storage classes in C?
auto, static, register, extern.

28. What is recursion?
Recursion is when a function calls itself.

29 . What is the base condition in recursion?
It is the condition that stops the recursive calls.

30. What is an infinite loop?
A loop that never stops because the condition is always true.

31. What is a nested loop?
A loop inside another loop.

32. What is a string in C?
A string is an array of characters ending with '\0'.

33. What is null character (\0)?
It indicates the end of a string.

34. What is the difference between gets() and fgets()?
`gets()` is unsafe (no limit checking).
`fgets()` is safer (limits input size).

35. What is the use of sizeof() operator?
It returns the size of a data type or variable in bytes.

36. What is type casting?
Converting one data type into another.
Example: `(float) a`

37. What is a macro?
A macro is a fragment of code defined using `#define`.

38. What is dynamic memory allocation?
Memory allocated during runtime using `malloc()`, `calloc()`, `realloc()`.

39. What is malloc()?
It allocates memory dynamically and returns a pointer.

40. What is free()?
It releases dynamically allocated memory.

41. What is a dangling pointer?
A pointer that points to memory that has been freed.

42. What is the difference between break and continue?
`break` exits loop.
`continue` skips current iteration.

43. What is enumeration (enum)?
A user-defined data type consisting of named integer constants.

44. What is the difference between structure and union?

Structure stores all members separately.
Union shares same memory for all members.

45. What is a function in C?

A function is a block of code that performs a specific task.

46. Why do we use functions?

To make the program modular, reusable, and easy to understand.

47. What is the syntax of a function?

```
return_type function_name(parameters)
{
    // body
}
```

48. What is function declaration?

It tells the compiler about the function before it is used.
Example: `int add(int, int);`

49. What is function definition?

It contains the actual code of the function.

50. What is function call?

It is the statement used to execute a function.Example: `add(a, b);`

51. What are parameters?

Variables listed in function definition to receive values.

52. What are arguments?

Actual values passed to the function during call.

53. What is the difference between parameters and arguments?

Parameters are variables in function definition.Arguments are values passed during function call.

54. What is return type of a function?

It specifies the type of value the function returns.

55. What is void function?

A function that does not return any value.

56. What is call by value?

Only a copy of the variable is passed. Original value does not change.

57. What is call by reference?

Address of variable is passed. Original value can change.

58. How is call by reference implemented in C?
Using pointers.

59. What is recursive function?
A function that calls itself.

60. What is base condition in recursion?
It is the condition that stops recursive calls.

61. What happens if base condition is missing?
The function will run infinitely and may cause stack overflow.

62. What is stack overflow?
It happens when too many function calls fill up memory stack.

63. What is function prototype?
It is another name for function declaration.

64.. What is the difference between library function and user-defined function?
Library functions are predefined (like printf()).
User-defined functions are created by the programmer.

65. What is function reusability?
A function can be used multiple times in a program.

66. What is modular programming?
Dividing a program into small functions/modules.

67. Why are functions important in large programs?
They make debugging, testing, and maintenance easier.

68. Why switch-case is used?
Because we check fixed menu choices (1–5).

69. Role of default case?
It handles invalid choices.

70. What if break is removed?
All cases after match will execute (fall-through).

71. Can switch work with float?
No, only int or char.