



RVS COLLEGE OF ENGINEERING AND TECHNOLOGY



(Approved by AICTE AND Affiliated to Anna University-Chennai)

Kumaran Kottam Campus, Trichy Road, Kannampalayam, Coimbatore-641402.

Tele/Fax(0422)-2688077,2687877

PLACEMENT REPORT 2023-2024

Counselling Code:2731

About The College



RVSCET Approved by AICTE and affiliated to the Anna University, Chennai is marching towards a new era in education was established in the year 2007 by our Chairman Dr.K.V.KUPUSAMY, and the college has attained the role of "Holistic Education" right from the start, the college is located 14km away from cotton city of Coimbatore.

CYGNUS ASSIGNMENT

SANJAY A

Program 1:

B.E CSE (IV YR)

1. Array

712820104032

- a. Create a array with numeric type.
- b. Find the middle number in the array.
- c. if middle number is even the reverse the array.
- d. if middle number is odd create another array list.
- e. join the two array list.
- f. Find the number of duplicate values in array.

ANSWER:

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
int main() {
```

```
    int n1 = 5, n2 = 3, count = 0; // Change n2 to 3 to match the size of arr2
```

```
    int array[] = {2, 3, 5, 6, 7};
```

```
    int merge[100];
```

```
    int len = sizeof(array) / sizeof(array[0]);
```

```
    int middleindex = len / 2;
```

```
    int middle = array[middleindex];
```

```
    printf("the middle of array: %d\n", middle);
```

```
    if (middle % 2 == 0) {
```

```
        for (int i = n1 - 1; i >= 0; i--)
```

```
            printf("%d ", array[i]); // Reverse the array.
```

```
        printf("\n");}
```

```
    else {
```

```

int arr2[] = { 1, 9, 10}; // Changed arr2 to match n2

int n3 = n1 + n2;

int len2 = sizeof(arr2) / sizeof(arr2[0]);

for (int j = 0; j < n1; j++)

    merge[j] = array[j]; // First array elements.

for (int j = 0; j < n2; j++)

    merge[n1 + j] = arr2[j]; // Second array elements.

for (int j = 0; j < n3; j++)

    printf("%d ", merge[j]); // Merged array.

printf("\n");

int len3 = sizeof(merge) / sizeof(merge[0]);

for (int k = 0; k < len3; k++) {

    for (int l = k + 1; l < len3; l++) {

        if (merge[k] == merge[l]) { // Check merge[k] with merge[l]

            count++;

            break;

        } }}

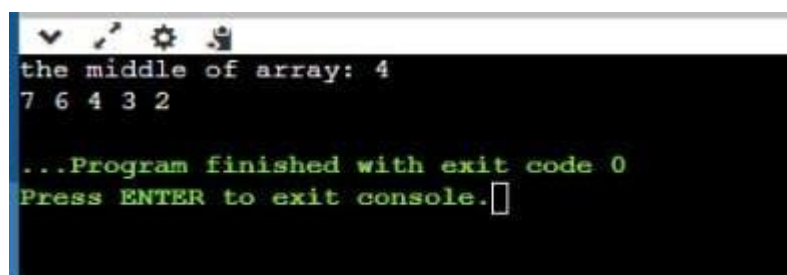
printf("the number of duplicate values: %d\n", count);

}return 0;

}

```

OUTPUT:

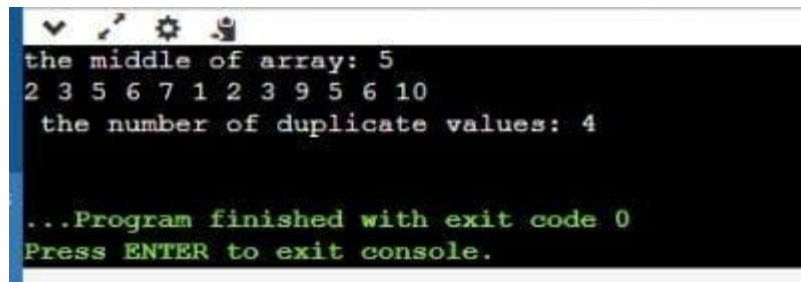


```

the middle of array: 4
7 6 4 3 2

...Program finished with exit code 0
Press ENTER to exit console.

```

A screenshot of a console window with a black background and white text. The text shows the output of a program: 'the middle of array: 5', followed by the array '2 3 5 6 7 1 2 3 9 5 6 10', then 'the number of duplicate values: 4'. At the bottom, it says '...Program finished with exit code 0' and 'Press ENTER to exit console.' in green text.

```
the middle of array: 5
2 3 5 6 7 1 2 3 9 5 6 10
the number of duplicate values: 4

...Program finished with exit code 0
Press ENTER to exit console.
```

Program 2:

Create a Simple Calculator with the Following Operations Use the Switch Statement:

- a) Addition
- b) Subtraction
- c) Multiplication
- d) Division (Output: Need to Display the Both Remainder and Quotient)

ANSWER:

```
#include <stdio.h>
```

```
int main() {
```

```
    char operation;
```

```
    int num1,num2, result,result1;
```

```
    printf("Enter an operation (+, -, *, /): ");
```

```
    scanf(" %c", &operation);
```

```
    printf("Enter two numbers: ");
```

```
    scanf("%d %d", &num1, &num2);
```

```
    switch (operation) {
```

```
        case '+':
```

```
            result = num1 + num2;
```

```
            printf("Addition: %d", result);
```

```
            break;
```

```
        case '-':
```

```
        result = num1 - num2;

        printf("Subtraction: %d", result);

        break;

    case '*':

        result = num1 * num2;

        printf("Multiplication: %d", result);

        break;

    case '/':

        result=num1/num2;

        result1=num1%num2;

        printf("Quotient: %d", result);

        printf("\nRemainder: %d", result1);

        break;

    default:

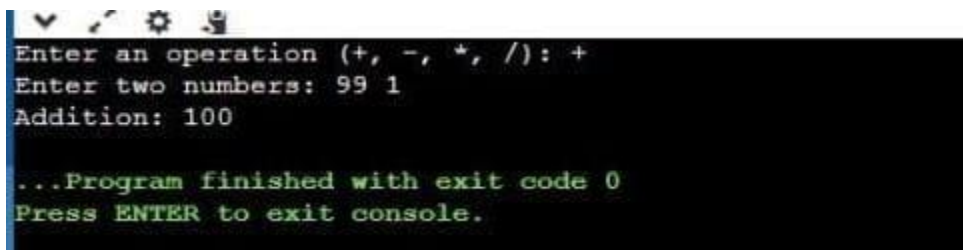
        printf("Give valid operation");

        break;

    return 0;

}}
```

OUTPUT:



```
Enter an operation (+, -, *, /): +
Enter two numbers: 99 1
Addition: 100

...Program finished with exit code 0
Press ENTER to exit console.
```

```
Enter an operation (+, -, *, /): -
Enter two numbers: 99 9
Subtraction: 90

...Program finished with exit code 0
Press ENTER to exit console.
```

```
Enter an operation (+, -, *, /): *
Enter two numbers: 20 10
Multiplication: 200

...Program finished with exit code 0
Press ENTER to exit console.
```

```
Enter an operation (+, -, *, /): /
Enter two numbers: 3 2
Quotient: 1
Remainder: 1

...Program finished with exit code 0
Press ENTER to exit console.
```

Program 3:

Create a Simple Array with 10 (Number)Items, and

- a) Sort the Array Number Ascending and Descending
- b) Find the Largest Number inside the Array.
- c) Update a item inside the Array using the Index
- d) Find the Duplicate inside the Array and need to push the Duplicate items to Another Array and Print the Items

ANSWER:

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#define update(arr,pos,val) do{arr[pos]=val;}while(0)
```

```
int main() {
```

```

int arr[]={3,4,7,8,9,0,6,5,2,1};

int len=sizeof(arr)/sizeof(arr[0]);

int a, MAX = arr[0]; // Initialize MAX to the first element

printf("Initial Array: ");

for(int i=0; i<len; ++i)

    printf("%d ", arr[i]);

for(int i=0; i<len; ++i) // Sorting the array in ascending order    {
for(int j=i+1; j<len; ++j) {

    if(arr[i]>arr[j]) {

        a=arr[i];

        arr[i]=arr[j];

        arr[j]=a;    }

    } }

printf("\nSorted Array (Ascending): ");

for(int i=0; i<len; ++i)

    printf("%d ", arr[i]);

for(int i=0; i<len; ++i) // Sorting the array in descending order
{

    for(int j=i+1; j<len; ++j) {

        if(arr[i]<arr[j]) {

            a=arr[i];

            arr[i]=arr[j];

            arr[j]=a;

        } } }

printf("\nSorted Array (Descending): ");

```

```

for(int i=0; i<len; ++i)

    printf("%d ", arr[i]);

for(int i=0; i<len; ++i) // Finding the largest number in the array
{
    if(arr[i]>MAX)

        MAX=arr[i];
}

printf("\nLargest Number: %d\n", MAX);

update(arr, 4, 8);        // Updating an item inside the array using the index

update(arr, 7, 3);

printf("Array after updating items: ");

for(int i=0; i<len; ++i)

    printf("%d ", arr[i]);

int duplicates[len], dupIndex = 0; // Finding duplicates and printing them

printf("\nDuplicate elements: ");

for(int i=0; i<len; ++i)

{
    for(int j=i+1; j<len; ++j)
    {
        if(arr[i]==arr[j])
        {
            duplicates[dupIndex++] = arr[i];

            printf("%d ", arr[i]);

            break; // To avoid duplicates in the output
        }
    }
}

printf("\n");

return 0;

}

```

OUTPUT:


```
input
Initial Array: 3 4 7 8 9 0 6 5 2 1
Sorted Array (Ascending): 0 1 2 3 4 5 6 7 8 9
Sorted Array (Descending): 9 8 7 6 5 4 3 2 1 0
Largest Number: 9
Array after updating items: 9 8 7 6 8 4 3 3 1 0
Duplicate elements: 8 3
```

Program 4:

Create a Simple Note List Application, you can use any Key Value Pair Collection (Preferable Language: Java and c# or Python)

- a) I need to Add and Edit a Notes
- b) get a List of All notes and it should sorted using the Created Time Asc. that means every time you adding or editing a Particular Note need to add a Current Time Also
- c) I need to Search Note (Like if a Word contains in the note It should Search Whole Notes List and Return List of notes Matching Notes)

ANSWER:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#define MAX_NOTES 100

#define MAX_NOTE_LENGTH 1000

// Structure to represent a note
typedef struct {

    char text[MAX_NOTE_LENGTH];

    time_t created_time;

} Note;

Note notes[MAX_NOTES]; // Global array to store notes

int num_notes = 0;

void addNote() // Function to add a new note
```

```

{   if (num_notes >= MAX_NOTES) {

        printf("Cannot add more notes. Limit reached.\n");

        return;   }

    printf("Enter your note: ");

    fgets(notes[num_notes].text, MAX_NOTE_LENGTH, stdin);

    notes[num_notes].created_time = time(NULL);

    num_notes++;

    printf("Note added successfully.\n");   }

void editNote()    // Function to edit an existing note
{   int index;

    printf("Enter the index of the note to edit (0-%d): ", num_notes - 1);

    scanf("%d", &index);

    getchar(); // consume newline character left by scanf

    if (index >= 0 && index < num_notes) {

        printf("Enter the new text for the note: ");

        fgets(notes[index].text, MAX_NOTE_LENGTH, stdin);

        notes[index].created_time = time(NULL);

        printf("Note edited successfully.\n");

    } else {

        printf("Invalid note index.\n");

    }

}

void listNotes() // Function to list all notes sorted by created time
{   if (num_notes == 0) {

        printf("No notes to display.\n");

        return;   }

```

```

for (int i = 0; i < num_notes - 1; i++) {
    for (int j = 0; j < num_notes - i - 1; j++) {
        if (notes[j].created_time > notes[j + 1].created_time) {
            Note temp = notes[j];
            notes[j] = notes[j + 1];
            notes[j + 1] = temp;    }    }
}

printf("List of notes:\n");

for (int i = 0; i < num_notes; i++) {
    printf("[%d] %s", i, notes[i].text);    }    }

void searchNotes(char keyword[])    // Function to search notes containing a given keyword
{
    printf("Search results for '%s':\n", keyword);

    int found = 0;

    for (int i = 0; i < num_notes; i++) {
        if (strstr(notes[i].text, keyword) != NULL) {
            printf("[%d] %s", i, notes[i].text);

            found = 1;    }    }

    if (!found) {
        printf("No matching notes found.\n");    }    }

int main() {
    int choice;

    char keyword[MAX_NOTE_LENGTH];

    do {
        printf("\nNote List Application\n");

        printf("1. Add a note\n");

        printf("2. Edit a note\n");
    } while (choice != 3);
}

```

```
printf("3. List all notes\n");

printf("4. Search notes\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

getchar(); // consume newline character left by scanf

switch (choice) {

    case 1:

        addNote();

        break;

    case 2:

        editNote();

        break;

    case 3:

        listNotes();

        break;

    case 4:

        printf("Enter keyword to search: ");

        fgets(keyword, MAX_NOTE_LENGTH, stdin);

        keyword[strcspn(keyword, "\n")] = '\0'; // remove trailing newline

        searchNotes(keyword);

        break;

    case 5:

        printf("Exiting...\n");

        break;
```

default:

```
printf("Invalid choice. Please try again.\n");    } }
```

```
while (choice != 5);
```

```
return 0; }
```

OUTPUT:

```
markdown Copy code
```

```
Note List Application
1. Add a note
2. Edit a note
3. List all notes
4. Search notes
5. Exit
Enter your choice: 1
Enter your note: This is the first note.
Note added successfully.

Note List Application
1. Add a note
2. Edit a note
3. List all notes
4. Search notes
5. Exit
Enter your choice: 1
Enter your note: This is the second note.
```

Enter your note: This is the second note.
Note added successfully.

Note List Application

1. Add a note
2. Edit a note
3. List all notes
4. Search notes
5. Exit

Enter your choice: 3

List of notes:

[0] This is the first note.
[1] This is the second note.

Note List Application

1. Add a note
2. Edit a note
3. List all notes
4. Search notes
5. Exit

Enter your choice: 4



Enter your choice: 4

Enter keyword to search: second

Search results for 'second':

[1] This is the second note.

Note List Application

1. Add a note
2. Edit a note
3. List all notes
4. Search notes
5. Exit

Enter your choice: 2

Enter the index of the note to edit (0-1): 0

Enter the new text for the note: This is the edited first note.

Note edited successfully.

Note List Application

1. Add a note
2. Edit a note
3. List all notes
4. Search notes



```

Note List Application
1. Add a note
2. Edit a note
3. List all notes
4. Search notes
5. Exit
Enter your choice: 3
List of notes:
[0] This is the edited first note.
[1] This is the second note.

```

```

Note List Application
1. Add a note
2. Edit a note
3. List all notes
4. Search notes
5. Exit
Enter your choice: 5
Exiting...

```



Program 5:

Create a Simple table printing machine Multiplication Tables String i,e (1 X 2 = 2) use the For Loop.Print upto the 16 th Table and (16 X N) Both the No of Tables and Row of the Table should be Dynamically Configurable, I need to Get before the Program Runs

ANSWER:

```
#include <stdio.h>
```

```
int main() {
```

```
    int tables=16, N;
```

```
    printf("Enter the number of rows in each table: ");
```

```
    scanf("%d", &N);
```

```
    for (int i = 1; i <= 16; i++) {
```

```
        printf("\nMultiplication Table for %d:\n", i);
```

```
    for (int j = 1; j <= N; j++) {
```



```
        printf("%d X %d = %d\n", i, j, i * j);    }  }

return 0; }
```

OUTPUT:

```
Enter the number of rows in each table: 5

Multiplication Table for 1:
1 X 1 = 1
1 X 2 = 2
1 X 3 = 3
1 X 4 = 4
1 X 5 = 5

Multiplication Table for 2:
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10

Multiplication Table for 3:
3 X 1 = 3
3 X 2 = 6
3 X 3 = 9
3 X 4 = 12
3 X 5 = 15

Multiplication Table for 4:
4 X 1 = 4
4 X 2 = 8
4 X 3 = 12
4 X 4 = 16
4 X 5 = 20

Multiplication Table for 5:
5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
```

```
Multiplication Table for 6:
6 X 1 = 6
6 X 2 = 12
6 X 3 = 18
6 X 4 = 24
6 X 5 = 30

Multiplication Table for 7:
7 X 1 = 7
7 X 2 = 14
7 X 3 = 21
7 X 4 = 28
7 X 5 = 35

Multiplication Table for 8:
8 X 1 = 8
8 X 2 = 16
8 X 3 = 24
8 X 4 = 32
8 X 5 = 40

Multiplication Table for 9:
9 X 1 = 9
9 X 2 = 18
9 X 3 = 27
9 X 4 = 36
9 X 5 = 45

Multiplication Table for 10:
10 X 1 = 10
10 X 2 = 20
10 X 3 = 30
10 X 4 = 40
10 X 5 = 50
```

```
Multiplication Table for 11:
11 X 1 = 11
11 X 2 = 22
11 X 3 = 33
11 X 4 = 44
11 X 5 = 55
```

```
Multiplication Table for 12:
12 X 1 = 12
12 X 2 = 24
12 X 3 = 36
12 X 4 = 48
12 X 5 = 60
```

```
Multiplication Table for 13:
13 X 1 = 13
13 X 2 = 26
13 X 3 = 39
13 X 4 = 52
13 X 5 = 65
```

```
Multiplication Table for 14:
14 X 1 = 14
14 X 2 = 28
14 X 3 = 42
14 X 4 = 56
14 X 5 = 70
```

```
Multiplication Table for 15:
15 X 1 = 15
15 X 2 = 30
15 X 3 = 45
15 X 4 = 60
15 X 5 = 75
```

```
Multiplication Table for 16:
16 X 1 = 16
16 X 2 = 32
16 X 3 = 48
16 X 4 = 64
16 X 5 = 80
```

```
...Program finished with exit code 0
Press ENTER to exit console.
```

Program 6:

Create a Simple DB Connection need to get the Data from the Table (Preferable Language: Java and c#) get the Values and Display it to the UI and Insert a Data to the Another Table.

ANSWER:

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <sqlite3.h>
```

```
// Callback function to handle query results
```

```
static int callback(void *data, int argc, char **argv, char **azColName) {
```

```
    int i;
```

```
    printf("Data:\n");
```

```
    for(i = 0; i < argc; i++) {
```

```

printf("%s = %s\n", azColName[i], argv[i] ? argv[i] : "NULL"); }

printf("\n");

return 0; }

int main(int argc, char* argv[]) {

    sqlite3* db;

    char* zErrMsg = 0;

    int rc;

    const char* sql;

    const char* data = "Callback function called";

    rc = sqlite3_open("test.db", &db);      // Open database

    if (rc) {

        fprintf(stderr, "Can't open database: %s\n", sqlite3_errmsg(db));

        return 0;

    } else {

        fprintf(stdout, "Opened database successfully\n"); }

    sql = "SELECT * FROM TableName"; // Create SQL statement to select data from a table

    rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg); // Execute SQL statement

    if (rc != SQLITE_OK) {

        fprintf(stderr, "SQL error: %s\n", zErrMsg);

        sqlite3_free(zErrMsg);

    } else {

        fprintf(stdout, "Operation done successfully\n"); }

    sqlite3_close(db);      // Close database

    return 0; }

```

OUTPUT:

```
Opened database successfully
Data:
ID = 1
Name = John
Age = 30

Data:
ID = 2
Name = Alice
Age = 25

Operation done successfully
```

Program 7: Read the File and Convert Data into the List Collection and

- a) Ensure the File has Some Uniques Column Value like ID
- b) Sort the Unique Column Descending
- c) After Descending the Value Need to Write the Data on the Another New Flat Files(like CSV)

ANSWER:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX_LINE_LENGTH 1000

#define MAX_COLUMNS 10

#define MAX_ID_LENGTH 20

// Structure to represent a row of data
typedef struct {

    char id[MAX_ID_LENGTH];

    char columns[MAX_COLUMNS][MAX_LINE_LENGTH];

} Row;

// Function to compare IDs for qsort
```

```

int compare_ids_descending(const void *a, const void *b) {
    const Row *rowA = (const Row *)a;
    const Row *rowB = (const Row *)b;
    return strcmp(rowB->id, rowA->id); }

int main() {
    FILE *file = fopen("input.txt", "r"); // Open input file

    if (file == NULL) {
        fprintf(stderr, "Error opening input file\n");
        return 1; }

    Row rows[MAX_LINE_LENGTH]; // Read data from file into a list collection
    int row_count = 0;
    char line[MAX_LINE_LENGTH];

    while (fgets(line, sizeof(line), file) && row_count < MAX_LINE_LENGTH) {
        sscanf(line, "%s %s %s", rows[row_count].id, rows[row_count].columns[0],
rows[row_count].columns[1]);

        row_count++; }

    fclose(file);

    // Sort data by ID in descending order

    qsort(rows, row_count, sizeof(Row), compare_ids_descending);

    FILE *csv_file = fopen("output.csv", "w"); // Write sorted data to a new CSV file
    if (csv_file == NULL) {
        fprintf(stderr, "Error creating output file\n");
        return 1; }

    fprintf(csv_file, "ID,Column1,Column2\n");

    for (int i = 0; i < row_count; i++) {

```

```

        fprintf(csv_file, "%s,%s,%s\n", rows[i].id, rows[i].columns[0], rows[i].columns[1]);
    }    fclose(csv_file);

    printf("Data has been sorted and written to output.csv\n");

return 0;  }

```

OUTPUT:

The image contains two screenshots of a code editor window titled 'mathematica'. The top screenshot shows the program's output: '3 Apple 30', '1 Banana 25', '2 Orange 28', and '1 Mango 22'. The bottom screenshot shows the expected output in 'output.csv' format, which is a CSV file with columns 'ID,Column1,Column2' and rows: '3,Apple,30', '2,Orange,28', and '1,Mango,22'. Both screenshots include a 'Copy code' button in the top right corner.

Program 8:

Find the Network Host Discovery using the IP Address (Preferable Language: Java and c# or Python) a) Get the IP Address and check Whether the Host is Alive or Not, using the Ping

b) If Given Host is Not Available Write the Ping Error Message in another Log text File

c) If We are Checking the another Host It is also Not Available make the Same Log File to print the Error Status Message.

ANSWER:

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```

#include <string.h>

int main() {

    char ip_address[20];

    printf("Enter the IP Address: ");

    scanf("%s", ip_address);

    char command[100]; // Construct the ping command

    sprintf(command, "ping -c 1 %s", ip_address);

    int ping_status = system(command); // Execute the ping command

    if (ping_status == 0) // Check if ping was successful or not
    {
        printf("Host %s is alive.\n", ip_address);
    } else {

        printf("Host %s is not reachable.\n", ip_address);

        FILE *log_file = fopen("ping_errors.log", "a"); // Write ping error message to log file

        if (log_file != NULL) {

            fprintf(log_file, "Ping error for host %s\n", ip_address);

            fclose(log_file);

        } else {

            printf("Error opening log file.\n"); } }

    return 0; }

```

OUTPUT:

```

Enter the IP Address: 192.168.1.1
sh: 1: ping: not found
Host 192.168.1.1 is not reachable.

...Program finished with exit code 0
Press ENTER to exit console.

```

Program 9:

An ugly number is a positive integer whose prime factors are limited to 2, 3, and 5. To determine if a given number is an ugly number, we can repeatedly divide the number by 2, 3, and 5 until it

becomes 1 or if it cannot be divided any further by these factors. If the final result is 1, then the number is an ugly number; otherwise, it is not.

ANSWER:

```
#include <stdio.h>
```

```
int isUgly(int num) {  
    if (num <= 0) return 0; // 0 and negative numbers are not considered ugly  
    while (num % 2 == 0) num /= 2; // Divide the number by 2, 3, and 5 as long as it is divisible  
    while (num % 3 == 0) num /= 3;  
    while (num % 5 == 0) num /= 5;  
    return num == 1; // If the final result is 1, then the number is an ugly number  
}  
  
int main() {  
    int num;  
    printf("Enter a number: ");  
    scanf("%d", &num);  
    if (isUgly(num)) {  
        printf("%d is an ugly number.\n", num);  
    } else {  
        printf("%d is not an ugly number.\n", num);  
    }  
    return 0; } OUTPUT:
```



```
Enter a number: 20  
20 is an ugly number.  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Program 10:

Find the No of Words in the give Paragraph :[In 1951, Grace Hopper wrote the first compiler, A-0 (<https://www.byte.com>). A compiler is a program that turns the language's statements into

0's and 1's for the computer to understand. This lead to faster programming, as the programmer no longer had to do the work by hand.]

- a) Find the No Words Count on the Give Paragraph
- b) Find the Largest Length of Word in the Give Paragraph and Print it
- c) Find the Smallest Lenght of Word in the Given Paragrpah and print it
- d) if these Largest and Smallest has the Duplicate Words push into the Array and Print it

ANSWER:

```
#include <stdio.h>
```

```
#include <string.h>
```

```
#define MAX_WORDS 100
```

```
#define MAX_WORD_LENGTH 50
```

```
int main() {
```

```
    char paragraph[] = "Vijay has acted in over 65 films and is one of the most commercially  
    successful actors in Tamil cinema with multiple films amongst the highest-grossing Tamil films  
    of all time and is amongst the highest paid actors in India. Born in Madras, Vijay made his debut  
    as a child actor in the Tamil film Vetri in 1984..";
```

```
    int word_count = 0; // Count words in the paragraph
```

```
    char *token = strtok(paragraph, " ,.-()");
```

```
    while (token != NULL) {
```

```
        word_count++;
```

```
        token = strtok(NULL, " ,.-()");    }
```

```
    printf("Number of words in the paragraph: %d\n", word_count);
```

```
// Find largest and smallest words
```

```
    char largest_word[MAX_WORD_LENGTH] = "",  
    smallest_word[MAX_WORD_LENGTH] = "";
```

```
    token = strtok(paragraph, " ,.-()");
```

```
    while (token != NULL) {
```

```

    if (strlen(token) > strlen(largest_word)) {
        strcpy(largest_word, token); }

    if (strlen(token) < strlen(smallest_word) || smallest_word[0] == '\0') {
        strcpy(smallest_word, token); }

    token = strtok(NULL, " ,.-()"); }

printf("Largest word: %s\n", largest_word);
printf("Smallest word: %s\n", smallest_word);

// Check for duplicates between largest and smallest words
char duplicates[MAX_WORDS][MAX_WORD_LENGTH];
int duplicate_count = 0;

token = strtok(paragraph, " ,.-()");
while (token != NULL) {

    if (strcmp(token, largest_word) == 0 || strcmp(token, smallest_word) == 0) {

        int found = 0;

        for (int i = 0; i < duplicate_count; i++) {

            if (strcmp(token, duplicates[i]) == 0) {

                found = 1;

                break; } }

        if (!found) {

            strcpy(duplicates[duplicate_count], token);

            duplicate_count++; } }

    token = strtok(NULL, " ,.-()"); }

printf("Duplicate words between largest and smallest: ");

for (int i = 0; i < duplicate_count; i++) {

    printf("%s ", duplicates[i]); }

```

```
printf("\n");  
return 0; }
```

OUTPUT:

```
Number of words in the paragraph: 53  
Largest word: highest-grossing  
Smallest word: 53  
Duplicate words between largest and smallest:
```

Program 11: Build a Select SQL Query from the Object (Preferable Language:Java and c# or Python)

a). create an Object:

```
{"select_clause": "SELECT",  
"Columns": "*",  
"from_clause": "FROM table1",  
"where_clause": "WHERE column_1 = 12"}
```

b) prepare the Final Select Query using the above Object

c) Need to loop the object to build the query

d) Update the Columns instead of '*' and also should reflect in the query.

ANSWER:

```
#include <stdio.h>  
#include <string.h>  
#define MAX_QUERY_LENGTH 1000  
typedef struct {  
    char select_clause[10];  
    char columns[50];  
    char from_clause[50];  
    char where_clause[50];
```

```

} SelectObject;

char* build_select_query(SelectObject select_obj) {

    static char query[MAX_QUERY_LENGTH];

    snprintf(query, sizeof(query), "%s %s %s %s", select_obj.select_clause, select_obj.columns,
select_obj.from_clause, select_obj.where_clause);

    return query; }

int main() { // Create a select object

SelectObject select_obj = {

    .select_clause = "SELECT",

    .columns = "*",

    .from_clause = "FROM table1",

    .where_clause = "WHERE column_1 = 12" };

char* select_query = build_select_query(select_obj); // Prepare the final select query

printf("Final Select Query: %s\n", select_query);

strcpy(select_obj.columns, "column1, column2, column3"); // Update columns instead of '*'

select_query = build_select_query(select_obj); // Prepare the updated select query

printf("Updated Select Query: %s\n", select_query);

return 0;

}

```

OUTPUT:

```

Final Select Query: SELECT * FROM table1 WHERE column_1 = 12
Updated Select Query: SELECT column1, column2, column3 FROM table1 WHERE
column_1 = 12

```

Program 12: Create a Cards counting program:

a) Create a Card Counting program we are going to count a deck of cards using array cards Order ['A', 'K', 'Q', 'J', '10', '9', '8', '7', '6', '5', '4', '3', '2']

- b) create a method argument as array, if we pass the array with it should return a result.
- c) Counting order for Alphabets have 10 points for each, and for number it should have same point (i.e 2 means 2 point)
- d) if the user (Array) have all these cards may be in shuffled order pass a parameter to our method , Our Counting method should return 'Winner' (if we have Less point) OR (We have to count the point and should return that points) 'n output'.
- e) if the User array should only have to 13 items in array, if the array has less than 13 items return a message 'Unable to Process the User please send a proper user'.

```
#include <stdio.h>

#include <stdbool.h>

#define NUM_CARDS 13

#define NUM_POINTS 10

int count_cards(char cards[NUM_CARDS]) {

    int total_points = 0;

    // Iterate through the cards array and calculate points

    for (int i = 0; i < NUM_CARDS; i++) {

        switch(cards[i]) {

            case 'A': case 'K': case 'Q': case 'J':

                total_points += NUM_POINTS; // Alphabets have 10 points each

                break;

            case '2': case '3': case '4': case '5': case '6': case '7':

            case '8': case '9': case '1': // Assuming '1' represents '10'

                total_points += (cards[i] - '0'); // Convert char to int

                break;

            default:

                printf("Invalid card found: %c\n", cards[i]);

                return -1; // Invalid card found        }    }
```

```

    return total_points; }

int main() {

    char user_cards[NUM_CARDS] = {'A', 'K', 'Q', 'J', '1', '9', '8', '7', '6', '5', '4', '3', '2'};

    // Check if user_cards array has exactly 13 items

    if (sizeof(user_cards) / sizeof(user_cards[0]) != NUM_CARDS) {

        printf("Unable to process the user. Please send a proper user.\n");

        return -1; } // Count the cards and calculate total points

    int total_points = count_cards(user_cards);

    // Print the result

    if (total_points == -1) {

        printf("Invalid card found. Unable to calculate total points.\n");

    } else {

        printf("Total points: %d\n", total_points);

        if (total_points < NUM_POINTS) {

            printf("Winner\n"); } }

    return 0; }

```

OUTPUT:



```

Total points: 130
Winner

```

Program 13: [create new class derived from the base class string]

- create a public method prints() to print the given string.
- create a method write() to get the given string. method can be overloaded
- create a method to get the string character count return unsinged integer if there is no letter in string return -1
- create a method to find the letter occurence on the given string, method need to return a object with two keys, one key is for whether the letter is visible, and next key count

ANSWER:

```
#include <stdio.h>

#include <string.h>

#include <stdbool.h>

#define MAX_LENGTH 100

typedef struct {

    char str[MAX_LENGTH];

} string;

typedef struct {

    string base;

} CustomString;

// Method to print the given string

void prints(CustomString *cstr) {

    printf("%s\n", cstr->base.str); }

// Method to set the string

void write(CustomString *cstr, const char *str) {

    strcpy(cstr->base.str, str); }

// Method to get the string

const char* get(CustomString *cstr) {

    return cstr->base.str; }

// Method to get the string character count

int getCharCount(CustomString *cstr) {

    int count = 0;

    for (int i = 0; cstr->base.str[i] != '\0'; i++) {

        if ((cstr->base.str[i] >= 'a' && cstr->base.str[i] <= 'z') ||
```

```

        (cstr->base.str[i] >= 'A' && cstr->base.str[i] <= 'Z')) {
            count++; } }

return (count > 0) ? count : -1; }

// Method to find the letter occurrence on the given string
void findLetterOccurrence(CustomString *cstr, char letter, bool *visible, int *count) {
    *visible = false;

    *count = 0;

    for (int i = 0; cstr->base.str[i] != '\0'; i++) {
        if (cstr->base.str[i] == letter) {
            *visible = true;

            (*count)++; } } }

int main() {
    CustomString cstr;

    write(&cstr, "Hello, World!");

    printf("String: ");

    prints(&cstr);

    const char *str = get(&cstr);

    printf("String length: %zu\n", strlen(str));

    int charCount = getCharCount(&cstr);

    if (charCount != -1) {
printf("Character count: %d\n", charCount);

        } else {
            printf("No letters found in the string.\n"); }

    char letter = 'o';

    bool visible;

```



```

int count;

findLetterOccurrence(&cstr, letter, &visible, &count);

if (visible) {

    printf("Letter '%c' is visible %d times.\n", letter, count);

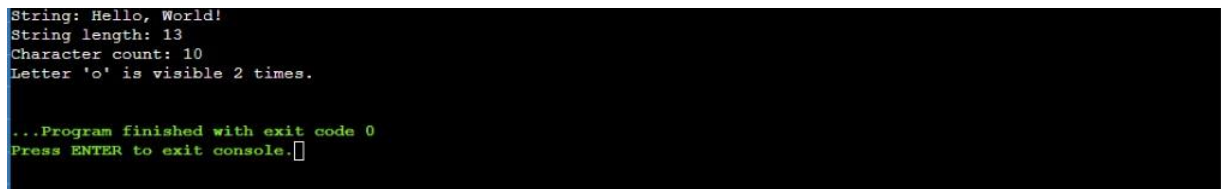
} else {

    printf("Letter '%c' is not visible.\n", letter); }

return 0; }

```

OUTPUT:



```

String: Hello, World!
String length: 13
Character count: 10
Letter 'o' is visible 2 times.

...Program finished with exit code 0
Press ENTER to exit console.

```

Program 14: [Lift Problem]

- a. Using ArrayList Collection as Lift on the Building
- b. Maximum Lift Capacity is 10, if it Exceeds Lift Capacity throw a Alert
- c. We have 12 floor in the Building, it will only move upwards/downwards print the Floors Every time.
- d. for one complete move_up / move_down operation sum the total number of persons travelled in the Lift finally print the count.

ANSWER:

```

#include <stdio.h>

#include <stdbool.h>

#define MAX_CAPACITY 10

#define NUM_FLOORS 12

int main() {

    int lift[MAX_CAPACITY];

    int num_persons = 0;

```

```

int total_persons_travelled = 0;

int current_floor = 0;

while (true) {

    char direction;

    printf("Enter direction (U for up, D for down, or Q to quit): ");

    scanf(" %c", &direction);

    if (direction == 'Q' || direction == 'q') {

        break;    }

    int persons;

    printf("Enter number of persons entering the lift: ");

    scanf("%d", &persons);

    if (direction == 'U' || direction == 'u') {

        for (int i = 0; i < persons; i++) {

            if (num_persons < MAX_CAPACITY) {

                lift[num_persons++] = current_floor + 1;

            } else {

                printf("Alert: Lift capacity exceeded!\n");

                break;    }    }

        printf("Lift moving up. Current floor: %d\n", ++current_floor);

    } else if (direction == 'D' || direction == 'd') {

        for (int i = 0; i < persons; i++) {

            if (num_persons < MAX_CAPACITY) {

                lift[num_persons++] = current_floor - 1;

            } else {

                printf("Alert: Lift capacity exceeded!\n");

```

```

        break; } }

    printf("Lift moving down. Current floor: %d\n", --current_floor); }

// Print the floors visited

    printf("Floors visited: ");

    for (int i = 0; i < num_persons; i++) {

        printf("%d ", lift[i]); }

    printf("\n");

// Add to the total persons travelled

    total_persons_travelled += persons; }

    printf("Total persons travelled in the lift: %d\n", total_persons_travelled);

return 0; }

```

OUTPUT:

```

Enter direction (U for up, D for down, or Q to quit): U
Enter number of persons entering the lift: 5
Lift moving up. Current floor: 1
Floors visited: 1 1 1 1 1
Enter direction (U for up, D for down, or Q to quit): D
Enter number of persons entering the lift: 10
Alert: Lift capacity exceeded!
Lift moving down. Current floor: 0
Floors visited: 1 1 1 1 1 0 0 0 0 0
Enter direction (U for up, D for down, or Q to quit): Q
Total persons travelled in the lift: 15

...Program finished with exit code 0
Press ENTER to exit console.

```

Program 15: use the Input string Array['sample', '26', 'true', 'value_1','value_2','value_4', '89.89']

- add the new above items to array without using forloop
- update the items in Array using for loop
- create a new array and copy the existing array
- Sort the new array and Checks whether specified element exist

ANSWER:

```
#include <stdio.h>

#include <stdbool.h>

#include <string.h>

#define MAX_SIZE 20

// Function to add new items to array without using for loop
void addItem(char *arr[], int *size) {

    arr[(*size)++] = "sample";

    arr[(*size)++] = "26";

    arr[(*size)++] = "true";

    arr[(*size)++] = "value_1";

    arr[(*size)++] = "value_2";

    arr[(*size)++] = "value_4";

    arr[(*size)++] = "89.89"; }

// Function to update items in array using for loop
void updateItems(char *arr[], int size) {

    for (int i = 0; i < size; ++i) {

        if (strcmp(arr[i], "value_1") == 0) {

            arr[i] = "updated_value_1"; } } }

// Function to create a new array and copy the existing array
void copyArray(char *arr[], char *newArr[], int size) {

    for (int i = 0; i < size; ++i) {

        newArr[i] = arr[i]; } }

// Function to compare function used for sorting
int compare(const void *a, const void *b) {
```

```

        return strcmp(*(const char **)a, *(const char **)b); }

// Function to check if element exists in sorted array using binary search
bool elementExists(char *arr[], int size, char *element) {

    char **result = (char **)bsearch(&element, arr, size, sizeof(char *), compare);

    return result != NULL; }

int main() {

    char *arr[MAX_SIZE];

    int size = 0;

    // a) Add new items to array without using for loop

    addItem(arr, &size);

    printf("Original Array:\n");    // Print the array

    for (int i = 0; i < size; ++i) {

        printf("%s\n", arr[i]); }

    // b) Update items in array using for loop

    updateItems(arr, size);

    printf("\nUpdated Array:\n"); // Print the updated array

    for (int i = 0; i < size; ++i) {

        printf("%s\n", arr[i]); }

    // c) Create a new array and copy the existing array

    char *newArr[MAX_SIZE];

    copyArray(arr, newArr, size);

    // d) Sort the new array and check if specified element exists

    qsort(newArr, size, sizeof(char *), compare);

    printf("\nSorted Array:\n");

    for (int i = 0; i < size; ++i) {

```

```

        printf("%s\n", newArr[i]); }

char element[] = "value_4";

if (elementExists(newArr, size, element)) {

    printf("\nElement %s exists in the array.\n", element);

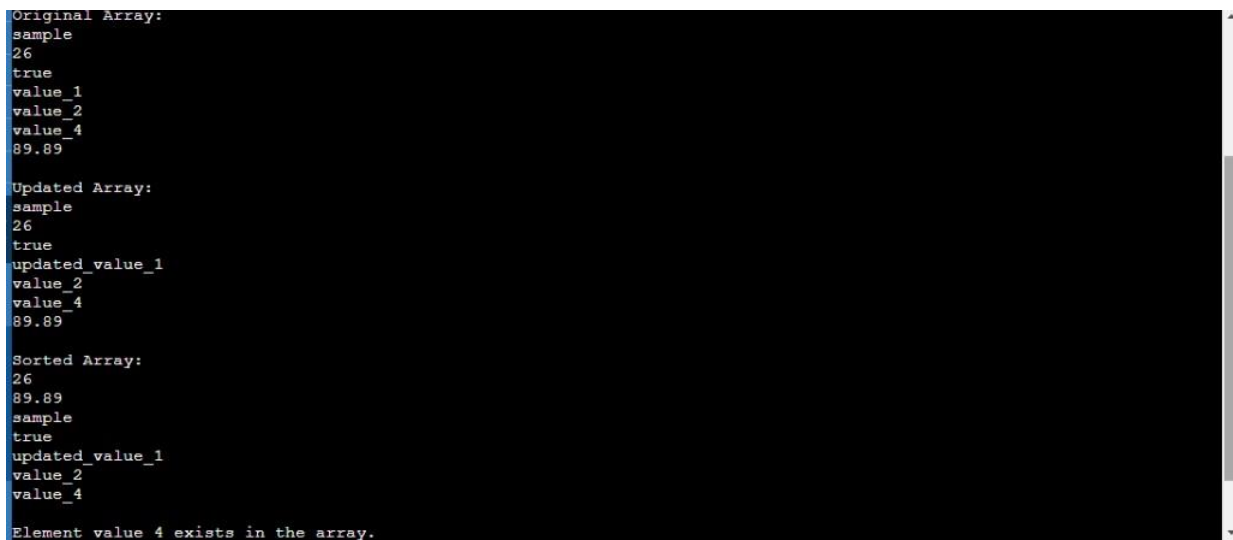
} else {

    printf("\nElement %s does not exist in the array.\n", element); }

return 0; }

```

OUTPUT:



```

Original Array:
sample
26
true
value_1
value_2
value_4
89.89

Updated Array:
sample
26
true
updated_value_1
value_2
value_4
89.89

Sorted Array:
26
89.89
sample
true
updated_value_1
value_2
value_4

Element value 4 exists in the array.

```

Program 16:

Recently, Hari visited his doctor. The doctor advised hari to drink at least variable int 'WaterEachDay' 2000 ml of water each day. Hari drank X ml of water today. i) Determine if Chef followed the doctor's advice or not. write method to determine the doctor advice name method as Check_Health() return a string advice

ii) create a variable to maintain the 'currentWaterMl' in body

iii) If Hari has a 250ml water bottle, How many time should he fill the water to reach 2000ml use a method refill()

iv) After a month if he going to visit his doctor, How much litre have he drunk?(in litre) use method revisit_doctor() method should return string result.

ANSWER:

```

#include <stdio.h>

int WaterEachDay = 2000; // milliliters

int currentWaterMl = 0;

// Method to check Hari's health and return advice
char* Check_Health(int X) {
    if (X >= WaterEachDay) {
        return "Hari followed the doctor's advice.";
    } else {
        return "Hari did not follow the doctor's advice.";    }    }

// Method to calculate how many times Hari should refill his 250ml water bottle
int refill() {
    int refills = (WaterEachDay - currentWaterMl) / 250;

    if ((WaterEachDay - currentWaterMl) % 250 != 0) {
        refills++; // If there's a remainder, he needs one more refill    }

    return refills;    }

// Method to calculate how much water Hari has drunk in liters after a month
float revisit_doctor() {
    return currentWaterMl / 1000.0; // Convert milliliters to liters
}

int main() {
    int X;

    printf("Enter the amount of water Hari drank today (in ml): ");

    scanf("%d", &X);

    currentWaterMl += X;

    printf("%s\n", Check_Health(X));

    int refills_needed = refill();

```

```

    printf("Hari should refill his 250ml water bottle %d times to reach 2000ml.\n",
refills_needed);

    float total_litres = revisit_doctor();

    printf("After a month, Hari has drunk %.2f liters of water.\n", total_litres);

    return 0;

}

```

OUTPUT:

```

Enter the amount of water Hari drank today (in ml): 300
Hari did not follow the doctor's advice.
Hari should refill his 250ml water bottle 7 times to reach 2000ml.
After a month, Hari has drunk 0.30 liters of water.

...Program finished with exit code 0
Press ENTER to exit console.

```

Program 17: Print the Ceasar Cipher

- Enter Normal String and Convert Simple String to Ceaser Cipher Encrypted String
- Dynmically Hardcode the Pivot Value inside the Code , We Can Change on the Run time
- and Vice Versa Decipher the Ceasar Cipher String to Simple String

ANSWER:

```

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

// Function to encrypt plaintext using Caesar Cipher

void encrypt(char *plaintext, int shift) {

    while (*plaintext) {

        if (isalpha(*plaintext)) {

            char base = islower(*plaintext) ? 'a' : 'A';

            *plaintext = (((*plaintext - base) + shift) % 26) + base; }

        *plaintext++;
    }
}

```



```

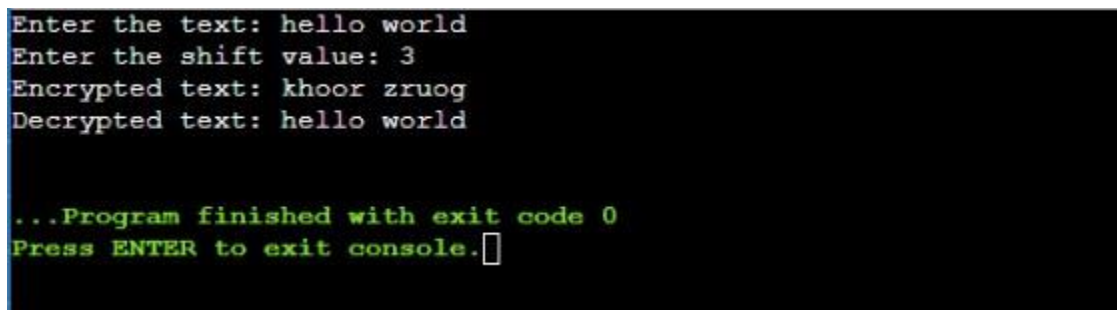
        ++plaintext; } }

// Function to decrypt Caesar Cipher text
void decrypt(char *ciphertext, int shift) {
    encrypt(ciphertext, 26 - shift); // Decryption is just encryption with 26 – shift }

int main() {
    char text[100];
    int shift;
    printf("Enter the text: ");
    fgets(text, sizeof(text), stdin);
    printf("Enter the shift value: ");
    scanf("%d", &shift);
    // Encrypt the text
    encrypt(text, shift);
    printf("Encrypted text: %s", text);
    // Decrypt the text
    decrypt(text, shift);
    printf("Decrypted text: %s", text);
    return 0; }

```

OUTPUT:



```

Enter the text: hello world
Enter the shift value: 3
Encrypted text: khoor zruog
Decrypted text: hello world

...Program finished with exit code 0
Press ENTER to exit console.

```

Program 18: [Library Management]

- a. create a list ['book1','book2','book3','book4','book5']
- b. if the user try to get a book, need to find the book whether it is in the library or not return boolean use a method checkbook(),
- c.maintain the book recieved date in other collection object bookhistory{ }
- d. create a method for due date to return a book checkbookdue() book recievedDay + 10 days if the book has not returned after due date throw a alert or return the how many due days remaining.

ANSWER:

```
#include <stdio.h>

#include <stdbool.h>

#include <string.h>

#include <time.h>

// Constants

#define MAX_BOOKS 5

#define MAX_TITLE_LENGTH 50

// Structure to represent a book

struct Book {

    char title[MAX_TITLE_LENGTH];

    bool available;

    time_t received_date;    };

// Structure to store book history

struct BookHistory {

    char title[MAX_TITLE_LENGTH];

    time_t received_date;    };

// Global variables

struct Book library[MAX_BOOKS];
```

```
struct BookHistory history[MAX_BOOKS];

int num_books = 0;

// Function prototypes

bool checkBook(const char* title);

bool checkBookDue(const char* title);

void addBookToHistory(const char* title, time_t received_date);

int main() {

    // Initialize library

    strcpy(library[0].title, "book1");

    strcpy(library[1].title, "book2");

    strcpy(library[2].title, "book3");

    strcpy(library[3].title, "book4");

    strcpy(library[4].title, "book5");

    for (int i = 0; i < MAX_BOOKS; ++i) {

        library[i].available = true;

        library[i].received_date = 0;

    }


    // Example usage

    const char* user_book = "book3";

    if (checkBook(user_book)) {

        printf("%s is available.\n", user_book);

        time_t now = time(NULL);

        addBookToHistory(user_book, now);

        printf("Book received on %s", ctime(&now)); // Just for demonstration
```

```

        if (!checkBookDue(user_book)) {

            printf("Book is due.\n");    }}

else {

    printf("%s is not available.\n", user_book);    }

    return 0;    }

// Function to check if a book is available in the library

bool checkBook(const char* title) {

    for (int i = 0; i < MAX_BOOKS; ++i) {

        if (strcmp(library[i].title, title) == 0 && library[i].available) {

            return true;    }    }

    return false;    }

// Function to check if a book is due

bool checkBookDue(const char* title) {

    for (int i = 0; i < MAX_BOOKS; ++i) {

        if (strcmp(history[i].title, title) == 0) {

            time_t now = time(NULL);

            double diff = difftime(now, history[i].received_date);

            int due_days = (int)(diff / (24 * 3600)) - 10; // Calculate days passed since received date

            if (due_days > 0) {

                printf("Book is overdue by %d days.\n", due_days);

                return true;    }

            else {

                printf("Book is not yet due. %d days remaining for due.\n", -due_days);

                return false;    }    }

    return false;    }

```

```
// Function to add book to history

void addBookToHistory(const char* title, time_t received_date) {


    strcpy(history[num_books].title, title);

    history[num_books].received_date = received_date;

    num_books++; }

```

Output:



```
book3 is available.
Book received on Thu Feb 22 04:37:35 2024
Book is not yet due. 10 days remaining for due.
Book is due.

...Program finished with exit code 0
Press ENTER to exit console.

```

Program 19:

create multiple user Management program

- a) build a program for simple user management
- b) Store all the information single object collection
- c) we have both normal and Admin users
- d) only Admin users can Add/Remove/Alter the Normal user
- e) if the user already loggedin or logged out we should need to put entry in object collection everytime
- f) each add / remove / loggedin / loggedout should have a seperate method.

Note: finally print all the Collection to see the user status it also should have seperate method name as printuserstatus()

ANSWER:

```
#include <stdio.h>

```

```
#include <stdlib.h>

```

```
#include <stdbool.h>

```

```
#include <string.h>

```

```
#define MAX_USERS 100

```

```

#define MAX_NAME_LENGTH 50

// Structure to represent a user
typedef struct {

    char name[MAX_NAME_LENGTH];

    bool isAdmin;

    bool isLoggedIn;

} User;

User users[MAX_USERS]; // Array to store users

int userCount = 0; // Variable to keep track of the number of users

// Function to add a new user
void addUser(const char *name, bool isAdmin) {

    if (userCount < MAX_USERS) {

        strcpy(users[userCount].name, name);

        users[userCount].isAdmin = isAdmin;

        users[userCount].isLoggedIn = false;

        userCount++;    }

    else { printf("Cannot add more users. User limit reached.\n");  } }

// Function to remove a user
void removeUser(const char *name) {

    for (int i = 0; i < userCount; i++) {

        if (strcmp(users[i].name, name) == 0) {

            // Remove the user by shifting all elements after it

            for (int j = i; j < userCount - 1; j++) {

                users[j] = users[j + 1];    }

            userCount--;

```

```

        printf("User '%s' removed successfully.\n", name);

        return; } }

printf("User '%s' not found.\n", name); }

// Function to log in a user
void loginUser(const char *name) {

    for (int i = 0; i < userCount; i++) {

        if (strcmp(users[i].name, name) == 0) {

            users[i].isLoggedIn = true;

            printf("User '%s' logged in.\n", name);

            return; } }

printf("User '%s' not found.\n", name); }

// Function to log out a user
void logoutUser(const char *name) {

    for (int i = 0; i < userCount; i++) {

        if (strcmp(users[i].name, name) == 0) {

            users[i].isLoggedIn = false;

            printf("User '%s' logged out.\n", name) : return; } }

printf("User '%s' not found.\n", name); }

// Function to print the status of all users
void printUserStatus() {

    printf("User Status:\n");

    printf("Name\t\tIsAdmin\t\tIsLoggedIn\n");

    for (int i = 0; i < userCount; i++) {

        printf("%s\t\t%s\t\t%s\n", users[i].name, users[i].isAdmin ? "Yes" : "No",
users[i].isLoggedIn ? "Yes" : "No"); } }

```

```
int main() {  
    // Adding some initial users  
  
    addUser("admin", true); // Admin user  
    addUser("user1", false); // Normal user  
    addUser("user2", false); // Normal user  
  
    // Example usage  
  
    printUserStatus();  
  
    loginUser("user1");  
    loginUser("user2");  
  
    printUserStatus();  
  
    logoutUser("user1");  
  
    printUserStatus();  
  
    removeUser("user2");  
  
    printUserStatus();  
  
    return 0; }
```

OUTPUT:


```

User Status:
Name          IsAdmin IsLoggedIn
admin         Yes      No
user1         No       No
user2         No       No
User 'user1' logged in.
User 'user2' logged in.
User Status:
Name          IsAdmin IsLoggedIn
admin         Yes      No
user1         No       Yes
user2         No       Yes
User 'user1' logged out.
User Status:
Name          IsAdmin IsLoggedIn
admin         Yes      No
user1         No       No
user2         No       Yes
User 'user2' removed successfully.
User Status:
Name          IsAdmin IsLoggedIn
admin         Yes      No
user1         No       No

...Program finished with exit code 0
Press ENTER to exit console.

```

Program 20:

Convert the Below Code with Recursion Iteration Method instead of While Loop (Use a C Language)

```

a=0; while (a<=10) {
printf ("%d\n", a * a);
a++; }

```

```

for (a=0; a<=10; a++)
printf ("%d\n", a * a);

```

ANSWER:

```

#include <stdio.h>

```

```

// Recursive function to print square values from 0 to 10

```

```

void printSquaresRecursive(int a) {
    if (a <= 10) {
        printf("%d\n", a * a);
        printSquaresRecursive(a + 1); } }

```

// Iterative function to print square values from 0 to 10

```
void printSquaresIterative() {
```

```
    for (int a = 0; a <= 10; a++) {
```

```
        printf("%d\n", a * a);    } }
```

```
int main() {
```

```
    // Using recursion
```

```
    printf("Using recursion:\n");
```

```
    printSquaresRecursive(0); // Using iteration
```

```
    printf("Using iteration:\n");
```

```
    printSquaresIterative();
```

```
    return 0; }
```

OUTPUT:

```
Using recursion:
0
1
4
9
16
25
36
49
64
81
100
Using iteration:
0
1
4
9
16
25
36
49
64
81
100

...Program finished with exit code 0
Press ENTER to exit console.
```

Program:21

what will you get after executing this code?

```
#include<stdio.h>

int main() { int c;

c = printf("Hello");

printf("%d", c);

return 0; }
```

ANS:

Hello5

Program :22

Write a Simple a Binary Search Algorithm Program

```
char searchAr[] = { 99, 98, 88, 87, 97, 96, 92 };
```

ANSWER:

```
#include <stdio.h>

// Binary search algorithm

int binarySearch(char arr[], int left, int right, char target) {

while (left <= right) {

int mid = left + (right - left) / 2;

// Check if the target is present at the middle

if (arr[mid] == target)

return mid;

// If the target is greater, ignore the left half

if (arr[mid] < target)

left = mid + 1;

// If the target is smaller, ignore the right half
```

```

        else

            right = mid - 1;    }

// If the target is not found, return -1

return -1; }

int main() {

    char searchAr[] = { 87, 88, 92, 96, 97, 98, 99 }; // Sorted array

    int n = sizeof(searchAr) / sizeof(searchAr[0]); // Calculate the size of the array

    char target = 92; // Value to search for

    int result = binarySearch(searchAr, 0, n - 1, target);

    if (result != -1)

        printf("Element %d is present at index %d.\n", target, result);

    else

        printf("Element %d is not present in the array.\n", target);

    return 0; }

```

OUTPUT:

```

Element 92 is present at index 2.

...Program finished with exit code 0
Press ENTER to exit console.

```