

LABWORK: 1

MIS: 612303017

NAME: AMAN BIPIN MORGHADE

Control Flow:

Qs.1)

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

```
int main() {
```

```
    int i = 3;
```

```
    while (i-->0) {
```

```
        int i = 100;
```

```
        i--;
```

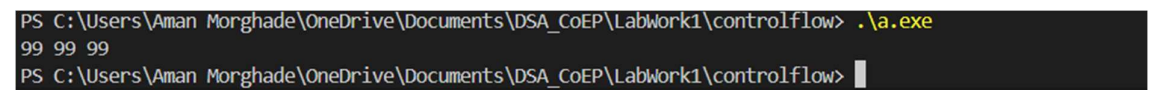
```
        printf("%d ", i);
```

```
    }
```

```
    return 0;
```

```
}
```

Output:



```
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\controlflow> .\a.exe
99 99 99
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\controlflow>
```

Qs 2)

```
int main() {
```

```
    float a, b;
```

```
    scanf("%f %f", &a, &b);
```

```
    int result = a + b;
```

```
    printf("%f %f %d", a, b, result);
```

```
    return 0;
```

```
}
```

OUTPUT:

```
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\controlflow> .\a.exe
20.1 5405
20.100000 5405.000000 5425
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\controlflow> █
```

Conditional Statements:

Write a C program to check whether a number is divisible by 5 and 11 or not.

Ans:

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

```
int main() {
    unsigned long int a;
    scanf("%d", &a);
    int check = a % 55 == 0 ? 1 : 0;
    if(check) {
        printf("Divisible\n");
    } else {
        printf("Not Divisible\n");
    };
    return 0;
};
```

OUTPUT:

```
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\conditional_statements> .\a.exe
11
Not Divisible
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\conditional_statements> .\a.exe
55
Divisible
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\conditional_statements> █
```

Qs.

Write a C program to input basic salary of an employee and calculate its Gross salary according to following: Basic Salary <= 10000 : HRA = 20%, DA = 80% Basic Salary <= 20000 : HRA = 25%, DA = 90% Basic Salary > 20000 : HRA = 30%, DA = 95%

Ans:

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

```

int main() {
    float salary;
    scanf("%f", &salary);
    if(salary <= 10000) {
        salary += 0.2 * salary + 0.8 * salary;
    } else if(salary <= 20000) {
        salary += 0.25 * salary + 0.9 * salary;
    } else {
        salary += 0.3 * salary + 0.95 * salary;
    };
    printf("%.3f", salary);
};

```

OUTPUT:

```

PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\conditional_statements> .\a.exe
55214.4
124232.398

```

Arrays:

Qs: Delete all duplicate elements from an array retaining the first occurrence. Note: Array elements cannot be deleted. shift and replace can be done.

Ans:

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

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

```

int main() {
    int size;
    printf("Enter Size of Array: ");
    scanf("%d", &size);
    int* arr = (int*) malloc(size * sizeof(int));
    for(int i = 0; i < size; i++) {
        scanf(" %d", &arr[i]);
    }
}

```

```

}

for(int i = 0; i < size; i++) {
    for(int j = i + 1; j < size; j++) {
        if (arr[i] == arr[j]) {
            for(int m = j; m < size - 1; m++) {
                arr[m] = arr[m + 1];
            };
            size--;
            j--;
        }
    }
}

for(int i = 0; i < size; i++) printf("%d ", arr[i]);

return 0;
}

```

OUTPUT:

```

PS C:\Users\Aman_Morghade\OneDrive\Documents\DSA_CoEP\Labwork1\arrays> .\a.exe
Enter Size of Array: 5
51 5 5 4 3
51 5 4 3
PS C:\Users\Aman_Morghade\OneDrive\Documents\DSA_CoEP\Labwork1\arrays>

```

Qs: Populate an array of size 100 with values generated randomly between 1 to 1000. Copy all the numbers divisible by 8 or 15 to a new array. Display both arrays.

Ans: #include <stdio.h>

#include <stdlib.h>

```

int main() {
    int arr[100];
    int result[100];
    int temp = 0;

```

```

for(int i = 0; i < 100; i++) {
    arr[i] = rand() % 1000 + 1;
};

for(int i = 0; i < 100; i++) {
    if(arr[i] % 8 == 0 || arr[i] % 15 == 0) result[temp++] = arr[i];
};

for(int i = 0; i < temp; i++) {
    printf("%d ", result[i]);
};
}

```

OUTPUT:

```

PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\arrays> .\a.exe
465 392 896 448 870 300 704 712 645 630 624 945 440 930

```

Strings:

Qs: Write a function to count number of occurrences of a character in a string;

Ans: #include <stdio.h>

#include <string.h>

```

int countOccurrencesofChar(char str[], char ch) {
    int len = strlen(str), count = 0;
    for(int i = 0; i < len; i++) {
        if(str[i] == ch) count++;
    };
    return count;
};

```

```

int main() {
    char str[100];
    char ch;

```

```

printf("Enter a string: ");

fgets(str, sizeof(str), stdin);

str[strcspn(str, "\n")] = '\0';

printf("Enter a character to count: ");

scanf("%c", &ch);

int count = countOccurrencesofChar(str, ch);

printf("The character '%c' occurs %d times in the string \"%s\".\n", ch, count, str);

return 0;
}

```

```

PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> gcc .\countOccurrence.c
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> .\a.exe
Enter a string: aman morghade
Enter a character to count: a
The character 'a' occurs 3 times in the string "aman morghade".
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string>

```

Qs:

* Write the strtok() function

Ans: #include <stdio.h>

#include <stdbool.h>

```

char* my_strtok(char* str, const char* delimiter) {

    static char* nextToken = NULL;

    if (str != NULL) {

        nextToken = str;

    }

    if (nextToken == NULL) {

        return NULL;

    }
}

```

```

char* start = nextToken;
while (*start && strchr(delimiter, *start)) {
    start++;
}

if (*start == '\0') {
    nextToken = NULL;
    return NULL;
}

char* end = start;
while (*end && !strchr(delimiter, *end)) {
    end++;
}

if (*end) {
    *end = '\0';
    nextToken = end + 1;
} else {
    nextToken = NULL;
}

return start;
}

```

Qs: Write a function which finds the longest possible subsequence of one string into another and returns the length + pointer to the subsequence.

Ans:

```

#include <stdio.h>

#include <string.h>

```

```
typedef struct {  
    int length;  
    char *ptr;  
} Result;
```

```
Result longestSubsequence(char *str1, char *str2) {  
    Result res;  
  
    res.length = 0;  
    res.ptr = NULL;  
  
    int len1 = strlen(str1);  
    int len2 = strlen(str2);  
  
    for (int i = 0; i < len1; i++) {  
        for (int j = 0; j < len2; j++) {  
            int k = 0;  
  
            while (i + k < len1 && j + k < len2 && str1[i + k] == str2[j + k]) {  
                k++;  
            }  
  
            if (k > res.length) {  
                res.length = k;  
                res.ptr = &str1[i];  
            }  
        }  
    }  
  
    return res;  
}
```

```
int main() {  
    char str1[] = "abcdemnopxyz";
```



```

char str2[] = "mnotq";

Result res = longestSubsequence(str1, str2);

printf("Length: %d, Subseq: %.*s\n", res.length, res.length, res.ptr);

return 0;
}

```

```

PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> gcc .\longestCommonSubsequence.c
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> .\a.exe
Length: 3, Subseq: mno
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string>

```

Qs: * Write a function to find gcd() of 2 numbers.

Ans: #include <stdio.h>

#include <string.h>

```

int gcd(int num1, int num2) {
    int temp;
    if(num1 <= num2) {
        temp = num1;
    } else {
        temp = num2;
    };

    for(int i = temp; i >= 1; i--) {
        if(num1 % i == 0 && num2 % i == 0) {
            return i;
        };
    };

    return 1;
}

// Time Complexity: O(n)
// Space complexity: O(1)

int main() {

```

```

int z, y;

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

int x = gcd(z, y);

printf("%d", x);

return 0;

}

```

```

PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> gcc .\gcd.c
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> .\a.exe
6 12
6
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> .\a.exe
98 15
1
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> .\a.exe
126 12
6

```

Qs: Write a function to find lcm() of 2 numbers

Ans: #include <stdio.h>

```

int find_lcm(int num1, int num2) {
    int temp1 = 2, temp2 = 2, flag = 1, result1 = num1, result2 = num2;
    while(flag) {
        if(result1 != result2) {
            if(result1 < result2) {
                result1 = num1*temp1;
                temp1++;
            } else {
                result2 = num2*temp2;
                temp2++;
            };
        } else {
            flag = 0;
            break;
        };
    };
};

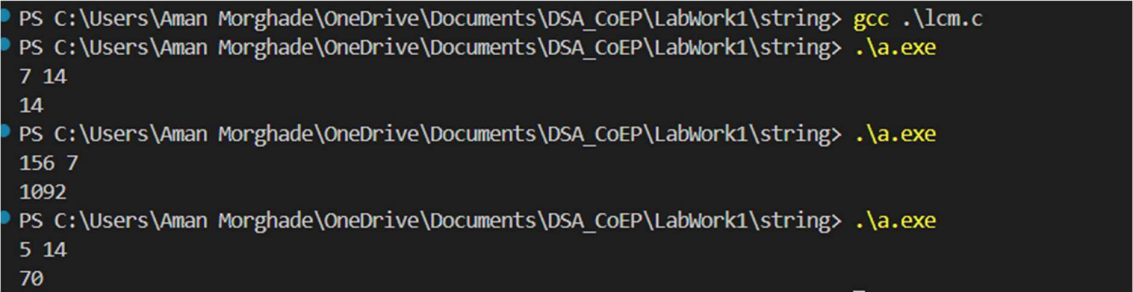
```

```

        return result1;
    };

int main() {
    int x, y;
    scanf("%d %d", &x, &y);
    int lcm = find_lcm(x, y);
    printf("%d", lcm);
    return 0;
}

```



```

PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> gcc .\lcm.c
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> .\a.exe
7 14
14
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> .\a.exe
156 7
1092
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> .\a.exe
5 14
70

```

Qs: Write a function to convert a decimal number to a binary number and return the binary representation in a string.

Ans: #include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX_SIZE 64

```

void reverse(char string[]) {
    int len = strlen(string), temp;
    for (int i = 0; i < len / 2; i++) {
        temp = string[i];
        string[i] = string[len - i - 1];
        string[len - i - 1] = temp;
    }
}

```

```
char* d2b(int num) {  
    char str[MAX_SIZE];  
    int len = 0;  
    if (num == 0) {  
        str[len++] = '0';  
    } else {  
        while (num) {  
            str[len++] = (num % 2) + '0';  
            num /= 2;  
        }  
    }  
    str[len] = '\\0';  
    reverse(str);  
  
    char* final = (char*)malloc(len + 1);  
    if (final != NULL) {  
        strcpy(final, str);  
    }  
  
    return final;  
}
```

```
int main() {  
    int x;  
    scanf("%d", &x);  
    char* binaryString = d2b(x);  
    printf("%s\\n", binaryString);  
    free(binaryString);  
    return 0;  
}
```

```
}
```

```
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> .\a.exe
17
10001
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> .\a.exe
123
1111011
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> .\a.exe
10
1010
```

Qs: * Write your own code for following library functions: strcasecmp strchr strcasecmp strcoll

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

```
#include <ctype.h>
```

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

```
int my_strcasecmp(const char *s1, const char *s2) {
```

```
    while (*s1 && *s2) {
```

```
        int diff = tolower(*s1) - tolower(*s2);
```

```
        if (diff != 0) {
```

```
            return diff;
```

```
        }
```

```
        s1++;
```

```
        s2++;
```

```
    }
```

```
    return *s1 - *s2;
```

```
}
```

```
char *my_strsep(char *str, const char *delim) {
```

```
    char *token = str;
```

```
    if (str == NULL) {
```

```
        return NULL;
```

```
    }
```

```
    if (*str == '\0') {
```

```
        return NULL;
```

```

    }

    while (*str && strchr(delim, *str) == NULL) {

        str++;

    }

    if (*str) {

        *str++ = '\0';

        while (*str && strchr(delim, *str)) {

            str++;

        }

    }

    return token;

}

```

```

int my_strcoll(const char *s1, const char *s2) {

    return strcmp(s1, s2);

}

```

Qs: *Write a program to reverse the digits of an integer and store the result as another integer

Ans: #include <stdio.h>

```

int rev(int num) {

    int rem, rev = 0;

    while(num) {

        rem = num%10;

        rev = rev*10 + rem;

        num /= 10;

    };

    return rev;

}

```

```

int main() {

```

```

int num;

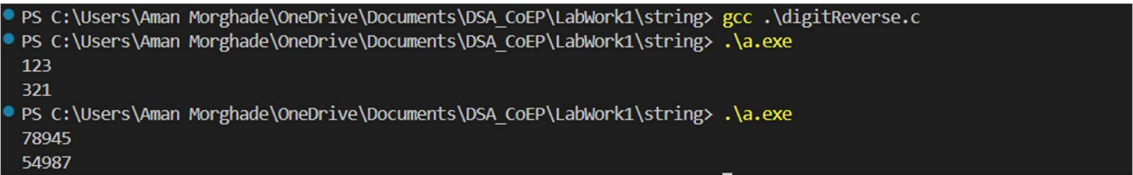
scanf("%d", &num);

printf("%d", rev(num));

return 0;

}

```



```

PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> gcc .\digitReverse.c
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> .\a.exe
123
321
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> .\a.exe
78945
54987

```

Qs: Write a program which reads a string and if the string has all digits in it, then derives the integer it represents.

Ans: #include <stdio.h>

#include <string.h>

```

int digitCheck(char p) {
    if(p >= '0' && p <= '9') {
        return 1;
    }
    return -1;
}

```

```

int main() {
    char digits[100];
    scanf("%s", digits);

    int len = strlen(digits);
    for(int i = 0; i < len; i++) {
        if(digitCheck(digits[i]) != 1) {
            printf("Non-digit character found: %c\n", digits[i]);
            return 0;
        }
    }
}

```

```

    }

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

        printf("%c", digits[i]);

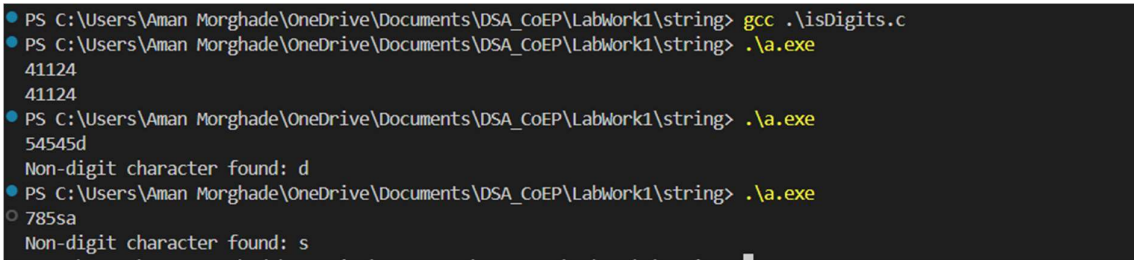
    }

    printf("\n");

    return 0;

}

```



```

PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> gcc .\isDigits.c
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> .\a.exe
41124
41124
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> .\a.exe
54545d
Non-digit character found: d
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\string> .\a.exe
785sa
Non-digit character found: s

```

Qs: Write a function which does the following void rev(char *str); Reverses the string "str" in place (without using another string).

Ans: #include <stdio.h>

#include <string.h>

```

void reverse(char string[]) {

    int len = strlen(string), temp;

    for(int i = 0; i < len / 2; i++) {

        temp = string[i];

        string[i] = string[len - i - 1];

        string[len - i - 1] = temp;

    }

    string[len] = '\0';

}

```

Qs: *Write a function which cuts a string given by "str" on the character given in "ch" and returns the first such word. char *cutonchar(char *str, char ch); For example: if 'str' is "something bad" and 'ch' is ' ' then it returns "something". if 'str' is "something bad" and 'ch' is 'e' then it returns 'som'

Ans: #include <string.h>

```
char* cutonchar(const char* str, char ch) {  
    const char* temp = str;  
    const char* closest_whitespace = NULL;  
    char* finalstr = NULL;  
    int len = 0;  
  
    while (*temp != '\0') {  
        if (*temp == ch) {  
            if (closest_whitespace == NULL) {  
                closest_whitespace = str;  
            }  
            len = closest_whitespace - str;  
            finalstr = (char*)malloc((len + 1) * sizeof(char));  
            if (finalstr == NULL) {  
                return NULL;  
            }  
            strncpy(finalstr, str, len);  
  
            finalstr[len] = '\0';  
            break;  
        } else if (*temp == ' ') {  
            closest_whitespace = temp + 1;  
        }  
        temp++;  
    }  
  
    return finalstr;  
}
```

Structures:

Q.1 and 2

Declare a structure that represents the following hierarchical information: to Remember (a) Student (b) Roll Number (c) Name (Typedef) (i) First name (ii) Middle Name (iii) Last Name (d) Gender (e) Date of Birth (Typedef) (i) Day (ii) Month (iii) Year (f) Marks (typedef) (i) English (ii) Mathematics (iii) Computer Science. Using the above structure, write a program to display the details of the student whose name is entered by the user. Display the name of the students who have secured less than 40% of the aggregate. In addition, print each student's average marks.

Ans: #include <stdio.h>

#include <string.h>

#include <stdlib.h>

```
typedef struct Name {  
    char* firstName;  
    char* middleName;  
    char* lastName;  
} Name;
```

```
typedef struct DateofBirth {  
    unsigned int day;  
    unsigned int month;  
    unsigned int year;  
} DateofBirth;
```

```
typedef struct Marks {  
    float English;  
    float Mathematics;  
    float ComputerScience;  
} Marks;
```

```
struct Student {  
    long int rollnumber;  
    Name* name;
```

```
char* gender;  
DateofBirth* dob;  
Marks* marks;  
};
```

```
void init(struct Student* s1) {  
    s1->name = (Name*)malloc(sizeof(Name));  
    s1->dob = (DateofBirth*)malloc(sizeof(DateofBirth));  
    s1->marks = (Marks*)malloc(sizeof(Marks));  
    s1->gender = (char*)malloc(10 * sizeof(char));  
}
```

```
void inputStudentDetails(struct Student* s) {  
    printf("Enter roll number: ");  
    scanf("%ld", &s->rollnumber);  
  
    printf("Enter first name: ");  
    s->name->firstName = (char*)malloc(50 * sizeof(char));  
    scanf("%s", s->name->firstName);  
  
    printf("Enter middle name: ");  
    s->name->middleName = (char*)malloc(50 * sizeof(char));  
    scanf("%s", s->name->middleName);  
  
    printf("Enter last name: ");  
    s->name->lastName = (char*)malloc(50 * sizeof(char));  
    scanf("%s", s->name->lastName);  
  
    printf("Enter gender (Male/Female): ");  
    scanf("%s", s->gender);
```

```

printf("Enter date of birth (DD MM YYYY): ");
scanf("%u %u %u", &s->dob->day, &s->dob->month, &s->dob->year);

printf("Enter marks in English: ");
scanf("%f", &s->marks->English);

printf("Enter marks in Mathematics: ");
scanf("%f", &s->marks->Mathematics);

printf("Enter marks in Computer Science: ");
scanf("%f", &s->marks->ComputerScience);
}

float calculateAverage(Marks* marks) {
    return (marks->English + marks->Mathematics + marks->ComputerScience) / 3.0;
}

void displayStudentDetails(struct Student* s) {
    printf("\nRoll Number: %ld\n", s->rollNumber);
    printf("Name: %s %s %s\n", s->name->firstName, s->name->middleName, s->name->lastName);
    printf("Gender: %s\n", s->gender);
    printf("Date of Birth: %02u/%02u/%04u\n", s->dob->day, s->dob->month, s->dob->year);
    printf("Marks: English: %.2f, Mathematics: %.2f, Computer Science: %.2f\n",
        s->marks->English, s->marks->Mathematics, s->marks->ComputerScience);
    printf("Average Marks: %.2f\n", calculateAverage(s->marks));
}

void displayStudentsWithLowAggregate(struct Student* students, int num_students) {
    printf("\nStudents with less than 40%% aggregate marks:\n");
    for(int i = 0; i < num_students; i++) {
        float average = calculateAverage(students[i].marks);
    }
}

```

```

        if(average < 40.0) {
            printf("%s %s %s - Average Marks: %.2f\n",
                students[i].name->firstName, students[i].name->middleName, students[i].name->lastName,
                average);
        }
    }
}

```

```

int main() {
    int num_students;

    printf("Enter the number of students: ");
    scanf("%d", &num_students);

    struct Student* students = (struct Student*)malloc(num_students * sizeof(struct Student));

    for(int i = 0; i < num_students; i++) {
        init(&students[i]);
        inputStudentDetails(&students[i]);
    }

    char firstName[50], middleName[50], lastName[50];
    printf("\nEnter the first name of the student to search: ");
    scanf("%s", firstName);
    printf("Enter the middle name of the student to search: ");
    scanf("%s", middleName);
    printf("Enter the last name of the student to search: ");
    scanf("%s", lastName);

    int found = 0;

    for(int i = 0; i < num_students; i++) {
        if(strcmp(students[i].name->firstName, firstName) == 0 &&

```

```
        strcmp(students[i].name->middleName, middleName) == 0 &&  
        strcmp(students[i].name->lastName, lastName) == 0) {  
            displayStudentDetails(&students[i]);  
            found = 1;  
            break;  
        }  
    }  
}
```

```
if(!found) {  
    printf("");  
}
```

```
displayStudentsWithLowAggregate(students, num_students);
```

```
for(int i = 0; i < num_students; i++) {  
    free(students[i].name->firstName);  
    free(students[i].name->middleName);  
    free(students[i].name->lastName);  
    free(students[i].name);  
    free(students[i].dob);  
    free(students[i].marks);  
    free(students[i].gender);  
}
```

```
free(students);
```

```
return 0;
```

```

}
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\structures> .\a.exe
Enter the number of students: 2
Enter roll number: 24
Enter first name: Aman
Enter middle name: Bipin
Enter last name: Morghade
Enter gender (Male/Female): Male
Enter date of birth (DD MM YYYY): 11 08 2005
Enter marks in English: 88
Enter marks in Mathematics: 95
Enter marks in Computer Science: 95
Enter roll number: 545
Enter first name: a
Enter middle name: b
Enter last name: c
Enter gender (Male/Female): f
Enter date of birth (DD MM YYYY): 22 07 2005
Enter marks in English: 87
Enter marks in Mathematics: 95
Enter marks in Computer Science: 95

Enter the first name of the student to search: a
Enter the middle name of the student to search: b
Enter the last name of the student to search: c

Roll Number: 545
Name: a b c
Gender: f
Date of Birth: 22/07/2005
Marks: English: 87.00, Mathematics: 95.00, Computer Science: 95.00
Average Marks: 92.33

Students with less than 40% aggregate marks:
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\structures>

```

Q.3 Write a program to define a structure for a hotel that has members— name, address, grade, number of rooms, and room charges. Write a function to print the names of hotels in a particular grade. Also write a function to print names of hotels that have room charges less than the specified value.

Ans: #include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct hotel {

 char* name;

 char* address;

 char grade;

 unsigned int numberOfRooms;

 unsigned int roomCharges;

} hotel;

void init(hotel* h) {

```
h->name = (char*)malloc(50 * sizeof(char));  
h->address = (char*)malloc(100 * sizeof(char));  
h->numberOfRooms = 0;  
h->roomCharges = 0;  
}
```

```
void printHotelsByGrade(hotel* hotels, int numHotels, char grade) {  
    printf("%c:\n", grade);  
    for (int i = 0; i < numHotels; i++) {  
        if (hotels[i].grade == grade) {  
            printf("- %s\n", hotels[i].name);  
        }  
    }  
}
```

```
void printHotelsByRoomCharges(hotel* hotels, int numHotels, unsigned int maxRoomCharges) {  
    printf("Hotels with room charges less than %u:\n", maxRoomCharges);  
    for (int i = 0; i < numHotels; i++) {  
        if (hotels[i].roomCharges < maxRoomCharges) {  
            printf("- %s\n", hotels[i].name);  
        }  
    }  
}
```

```
int main() {  
    int numHotels;  
  
    printf("Enter the number of hotels: ");  
    scanf("%d", &numHotels);  
  
    hotel* hotels = (hotel*)malloc(numHotels * sizeof(hotel));
```



```
for (int i = 0; i < numHotels; i++) {  
    init(&hotels[i]);  
    printf("\nEnter details for hotel %d:\n", i + 1);  
    printf("Name: ");  
    scanf("%[^\n]*c", hotels[i].name);  
    printf("Address: ");  
    scanf("%[^\n]*c", hotels[i].address);  
    printf("Grade: ");  
    scanf(" %c", &hotels[i].grade);  
    printf("Number of rooms: ");  
    scanf("%u", &hotels[i].numberOfRooms);  
    printf("Room charges: ");  
    scanf("%u", &hotels[i].roomCharges);  
}
```

```
char grade;  
printf("\nEnter the grade to search for hotels: ");  
scanf(" %c", &grade);  
printHotelsByGrade(hotels, numHotels, grade);
```

```
unsigned int maxRoomCharges;  
printf("\nEnter the maximum room charges ");  
scanf("%u", &maxRoomCharges);  
printHotelsByRoomCharges(hotels, numHotels, maxRoomCharges);
```

```
for (int i = 0; i < numHotels; i++) {  
    free(hotels[i].name);  
    free(hotels[i].address);  
}
```

```
free(hotels);
```

```

return 0;
}

```

```

PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\Labwork1\structures> gcc .\Qs3.c
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\Labwork1\structures> .\a.exe
Enter the number of hotels: 2

Enter details for hotel 1:
Name: Blossom
Address: ABC
Grade: A
Number of rooms: 230
Room charges: 2300

Enter details for hotel 2:
Name: The Inn
Address: XYZ
Grade: C
Number of rooms: 100
Room charges: 1000

Enter the grade to search for hotels: A
A:
- Blossom

Enter the maximum room charges 10000
Hotels with room charges less than 10000:
- Blossom
- The Inn
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\Labwork1\structures>

```

Q.4 Declare a structure time that has three fields—hr, min, sec. Create two variables start_time and end_time. Input their values from the user. Then while start_time does not reach the end_time, display GOOD DAY on the screen.

Ans: #include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <windows.h>

```
typedef struct {
```

```
    int hr;
```

```
    int min;
```

```
    int sec;
```

```
} Time;
```

```
int timeToSeconds(Time t) {
```

```
    return t.hr * 3600 + t.min * 60 + t.sec;
}
```

```
int main() {
    Time t1, t2;
    int temp1, temp2, temp3;

    printf("Enter First Time (Hr Min Sec): ");
    scanf("%d %d %d", &temp1, &temp2, &temp3);
    t1.hr = temp1;
    t1.min = temp2;
    t1.sec = temp3;

    printf("Enter Second Time (Hr Min Sec): ");
    scanf("%d %d %d", &temp1, &temp2, &temp3);
    t2.hr = temp1;
    t2.min = temp2;
    t2.sec = temp3;

    int start_time = timeToSeconds(t1);
    int end_time = timeToSeconds(t2);
    int diff = end_time - start_time;

    if (diff < 0) {
        printf("End time is before start time.\n");
        return 1;
    }

    for (int i = 0; i < diff; i++) {
        printf("GOOD DAY\n");
        sleep(1);
    }
}
```

```

    }

    return 0;
}

```

```

PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\structures> gcc .\Qs4.c
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\structures> .\a.exe
Enter First Time (Hr Min Sec): 10 15 50
Enter Second Time (Hr Min Sec): 18 57 04
GOOD DAY
GOOD DAY
GOOD DAY
GOOD DAY
GOOD DAY
GOOD DAY
GOOD DAY
GOOD DAY
GOOD DAY
GOOD DAY

```

Q.5 Declare a structure fraction that has two fields— numerator and denominator. Create two variables and compare them using function. Return 0 if the two fractions are equal, -1 if the first fraction is less than the second and 1 otherwise. You may convert a fraction into a floating point number for your convenience.

Ans: #include <stdio.h>

```

typedef struct fraction {
    float numerator;
    float denominator;
} fraction;

```

```

int compareFractions(float num1, float num2) {
    if(num1 == num2) {
        return 0;
    } else if(num1 < num2) {
        return -1;
    } else {
        return 1;
    }
};

```

```

int main() {

```

```

struct fraction f1, f2;

printf("First Fraction (Numerator Denominator): ");

scanf("%d %d", &f1.numerator, &f1.denominator);

printf("Second Fraction (Numerator Denominator): ");

scanf("%d %d", &f2.numerator, &f2.denominator);

if(f1.denominator == 0 || f2.denominator == 0) {

    printf("Invalid");

    return 0;

};

int result = compareFractions(f1.numerator / f1.denominator, f2.numerator / f2.denominator);

printf("%d", result);

return 0;

}

```

```

PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\structures> gcc .\Qs5.c
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\structures> .\a.exe
First Fraction (Numerator Denominator): 5 7
Second Fraction (Numerator Denominator): 14 15
-1
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\structures> .\a.exe
First Fraction (Numerator Denominator): 7 5
Second Fraction (Numerator Denominator): 1 7
1
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\structures> .\a.exe
First Fraction (Numerator Denominator): 5 7
Second Fraction (Numerator Denominator): 10 14
0
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\structures>

```

Q.6 Define a structure date containing three integers— day, month, and year. Write a program using functions to read data, to validate the date entered by the user and then print the date on the screen. For example, if you enter 29/2/2010 then that is an invalid date as 2010 is not a leap year. Similarly 31/6/2007 is invalid as June does not have 31 days.

Ans: #include <stdio.h>

#include <stdbool.h>

```

struct date {

    int day;

    int month;

    int year;

```

```
};
```

```
bool is_leap_year(int year) {  
    return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);  
}
```

```
bool is_valid_date(struct date d) {  
    int days_in_month[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};  
  
    if (d.month < 1 || d.month > 12) {  
        return false;  
    }
```

```
    if (d.month == 2 && is_leap_year(d.year)) {  
        return d.day >= 1 && d.day <= 29;  
    } else if (d.month == 2) {  
        return d.day >= 1 && d.day <= 28;  
    }
```

```
    return d.day >= 1 && d.day <= days_in_month[d.month];  
}
```

```
void read_date(struct date *d) {  
    printf("DD MM YYYY: ");  
    scanf("%d %d %d", &d->day, &d->month, &d->year);  
}
```

```
int main() {  
    struct date date;  
    read_date(&date);
```

```

    if (is_valid_date(date)) {
        printf("Valid\n");
    } else {
        printf("Invalid\n");
    }

    return 0;
}

```

```

PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\Labwork1\structures> gcc .\Qs6.c
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\Labwork1\structures> .\a.exe
DD MM YYYY: 27 07 2005
Valid
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\Labwork1\structures> .\a.exe
DD MM YYYY: 29 02 2020
Valid
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\Labwork1\structures> .\a.exe
DD MM YYYY: 12 21 2006
Invalid
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\Labwork1\structures> 

```

Q.7 Write a program to add and subtract 10hrs 20mins 50sec and 5hrs 30min 40sec.

Ans: #include <stdio.h>

```

struct time {
    int hr;
    int min;
    int sec;
};

```

```

void add_time(int hr1, int min1, int sec1, int hr2, int min2, int sec2) {
    int time1 = 3600 * hr1 + 60 * min1 + sec1;
    int time2 = 3600 * hr2 + 60 * min2 + sec2;
    int total_seconds = time1 + time2;

    int final_hr = total_seconds / 3600;
    total_seconds %= 3600;
}

```

```

    int final_min = total_seconds / 60;
    int final_sec = total_seconds % 60;

    printf("Added: %dhrs %dmins %dsec\n", final_hr, final_min, final_sec);
}

void subtract_time(int hr1, int min1, int sec1, int hr2, int min2, int sec2) {
    int time1 = 3600 * hr1 + 60 * min1 + sec1;
    int time2 = 3600 * hr2 + 60 * min2 + sec2;
    int total_seconds = time1 - time2;

    if (total_seconds < 0) {
        total_seconds = 0;
    }

    int final_hr = total_seconds / 3600;
    total_seconds %= 3600;
    int final_min = total_seconds / 60;
    int final_sec = total_seconds % 60;

    printf("Subtracted: %dhrs %dmins %dsec\n", final_hr, final_min, final_sec);
}

int main() {
    struct time t1, t2;

    printf("Time 1 (HH MM SS): ");
    scanf("%d %d %d", &t1.hr, &t1.min, &t1.sec);
    printf("Time 2 (HH MM SS): ");
    scanf("%d %d %d", &t2.hr, &t2.min, &t2.sec);

    add_time(t1.hr, t1.min, t1.sec, t2.hr, t2.min, t2.sec);
}

```



```
subtract_time(t1.hr, t1.min, t1.sec, t2.hr, t2.min, t2.sec);
```

```
return 0;
```

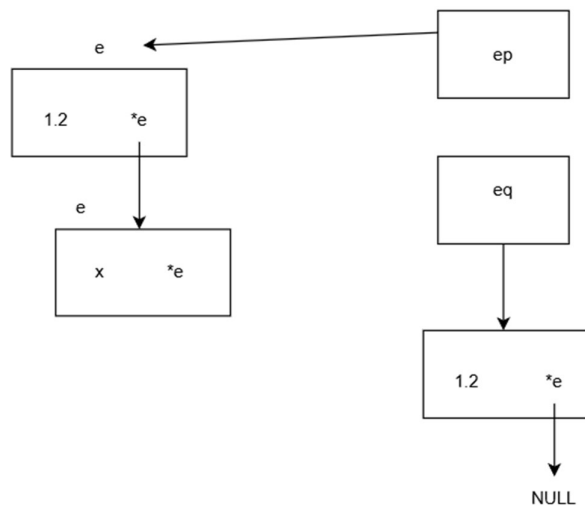
```
}
```

```
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\structures> gcc .\Qs7.c
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\structures> .\a.exe
Time 1 (HH MM SS): 10 20 50
Time 2 (HH MM SS): 17 54 17
Added: 28hrs 15mins 7sec
Subtracted: 0hrs 0mins 0sec
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\structures> .\a.exe
Time 1 (HH MM SS): 14 57 22
Time 2 (HH MM SS): 10 54 07
Added: 25hrs 51mins 29sec
Subtracted: 4hrs 3mins 15sec
PS C:\Users\Aman Morghade\OneDrive\Documents\DSA_CoEP\LabWork1\structures> |
```

Pointers:

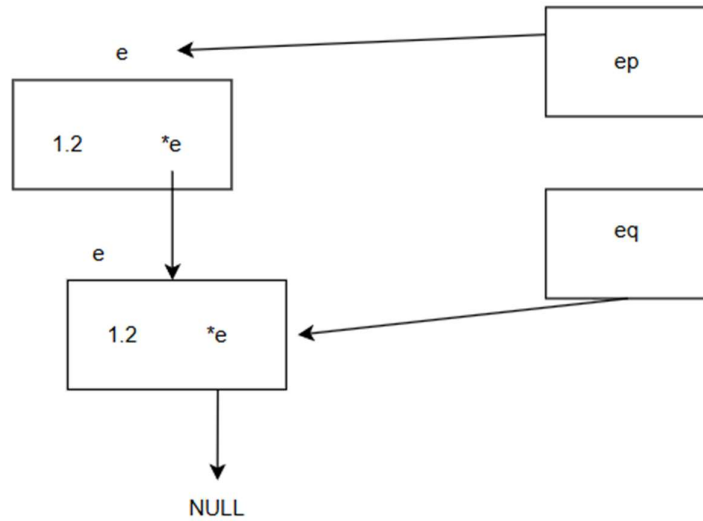
- 1) typedef struct entry { double x; struct entry *e; }entry; entry *f(entry **epp) { entry *ep = *epp; ep = (entry *)malloc(sizeof(entry)); ep->e = NULL; ep->x = 1.2; return ep; } int main() { entry e, *ep, *eq; ep = &e; e.e = (entry *)malloc(sizeof(entry)); eq = f(&(e.e)); e.x = eq->x; }

Qs_1)



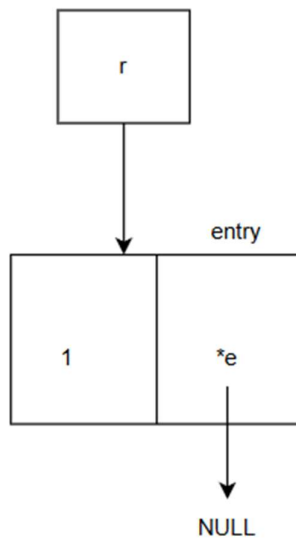
- 2) typedef struct entry { double x; struct entry *e; }entry; entry *f(entry **epp) { entry *ep = *epp; *epp = (entry *)malloc(sizeof(entry)); ep->e = NULL; ep->x = 1.2; return *epp; } int main() { entry e, *ep, *eq; ep = &e; e.e = (entry *)malloc(sizeof(entry)); eq = f(&(e.e)); e.x = eq->x; }

Qs_2)



- 3) `typedef struct entry { double x; struct entry *e; }entry; typedef entry *seq; void f(seq *p) { int i; seq r; *p = (seq) malloc(sizeof(entry)); r = *p; r->x = 100.0; for(i = 0; i < 2; i++) { r = (seq) malloc(sizeof(entry)); (*p)->e = r; r->e = NULL; r->x = (double) i; (*p) = (*p)->e; } } int main() { entry *r; f(&r); r = r->e; }`

Qs_3



POLYNOMIAL:

poly.h file:

```

typedef struct term {
    int coeff;
    int exp;
}
  
```

```

} term;

typedef struct polynomial {
    int n;
    struct term* t;
} poly;

void init_poly(poly* p, int size);
void append(poly* p, int coeff, int exp);
void display(poly* p);
void add_poly(poly* p1, poly* p2, poly* p3);
void sub_poly(poly* p1, poly* p2, poly* p3);
void quadratic_roots(poly* p);

```

poly.c file:

```

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "poly.h"

void init_poly(poly* p, int size) {
    p->n = 0;
    p->t = (term*) malloc(sizeof(term) * size);
    if(!p->t) return;
    return;
}; // The pointer is pointing to a dynamically allocated memory with size = sizeof(term) in
bytes

void append(poly* p, int coeff, int exp) {
    p->t[p->n].coeff = coeff;
    p->t[p->n].exp = exp;
    p->n++;
    return;
}; // The current memory address is added with a coefficient and exponent taken front the
user and the length of the polynomial is increased by 1

void display(poly* p) {
    int i = 0;
    for(i = 0; i < p->n - 1; i++) {
        printf("%dx^%d + ", p->t[i].coeff, p->t[i].exp);
    };
    printf("%dx^%d", p->t[i].coeff, p->t[i].exp);
    return;
} /** Each element is iterated over and displayed in an appropriate format
    Time Complexity: O(n)
    Space Complexity: O(1)
**/

```

```
void add_poly(poly* p1, poly* p2, poly* p3) { // This considers no repetition of same exponents within a single polynomial
```

```
    int i = 0;
    while(i < p1->n) {
        int helper = p1->t[i].exp;
        for(int j = 0; j < p2->n; j++) {
            if(p2->t[j].exp == helper) {
                p3->t[p3->n].coeff = p1->t[i].coeff + p2->t[j].coeff;
                p3->t[p3->n].exp = helper;
                p3->n++;
            };
        };
        i++;
    };
}
```

```
/** The exponents of the polynomial are matched using nested for loop and the coefficients are then added respectively
```

```
    Time Complexity:  $O(n^2)$ 
```

```
    Space Complexity:  $O(n)$ 
```

```
*/
```

```
void sub_poly(poly* p1, poly* p2, poly* p3) { // This considers no repetition of same exponents within a single polynomial
```

```
    int i = 0;
    while(i < p1->n) {
        int helper = p1->t[i].exp;
        for(int j = 0; j < p2->n; j++) {
            if(p2->t[j].exp == helper) {
                p3->t[p3->n].coeff = p1->t[i].coeff - p2->t[j].coeff;
                p3->t[p3->n].exp = helper;
                p3->n++;
            };
        };
        i++;
    };
}
```

```
/** The exponents of the polynomial are matched using a nested for loop and the coefficients are added respectively
```

```
    Time Complexity:  $O(n^2)$ 
```

```
    Space Complexity:  $O(n)$ 
```

```
*/
```

```
int isValidQuadratic(poly* p) {
```

```
    for(int i = 0; i < p->n; i++) {
        if(p->t[i].exp > 2) {
            return 0;
        };
    };
    return 1;
}
```

```
}; /** This functions checks if there is any element in the strucutre with exponent greater than 2, which breaks the quadratic of being quadratic */
```

```
void quadratic_roots(poly* p) { // standard equation:  $ax^2 + bx + c = 0$ 
    if(isValidQuadratic(p) == 0) return;

    int a = 0, b = 0, c = 0;
    for(int i = 0; i < p->n; i++) {
        if(p->t[i].exp == 2) {
            a = p->t[i].coeff;
        } else if(p->t[i].exp == 1) {
            b = p->t[i].coeff;
        } else if(p->t[i].exp == 0) {
            c = p->t[i].coeff;
        }
    }

    double discriminant = pow(b, 2) - 4 * a * c;

    if(discriminant < 0) {
        return;
    }

    double root1 = (-b + sqrt(discriminant)) / (2 * a);
    double root2 = (-b - sqrt(discriminant)) / (2 * a);
    printf("%lf %lf\n", root1, root2);

    return;
} /** The roots are calculated using the regular quadratic formula with constant time and
space complexity
Time Complexity: O(1)
Space Complexity: O(1)
**/
```

main.c file:

```
#include <stdio.h>
#include <stdlib.h>
#include "poly.c"
```

```
int main() {
    poly* p, *p2, *p3;
    p = (poly*) malloc(sizeof(p));
    p2 = (poly*) malloc(sizeof(p));
    p3 = (poly*) malloc(sizeof(p));
    init_poly(p, 3);
    init_poly(p2, 3);
    init_poly(p3, 3);
    append(p, 4, 2);
    append(p, 7, 1);
```

```
    append(p, 3, 0);
    append(p2, 2, 2);
    append(p2, 7, 1);
    add_poly(p, p2, p3);
    sub_poly(p, p2, p3);
    printf("\n");
    quadratic_roots(p);
    return 0;
}
```

-----**END**-----