

CDS IMPORTANT QUESTIONS for MID SEM

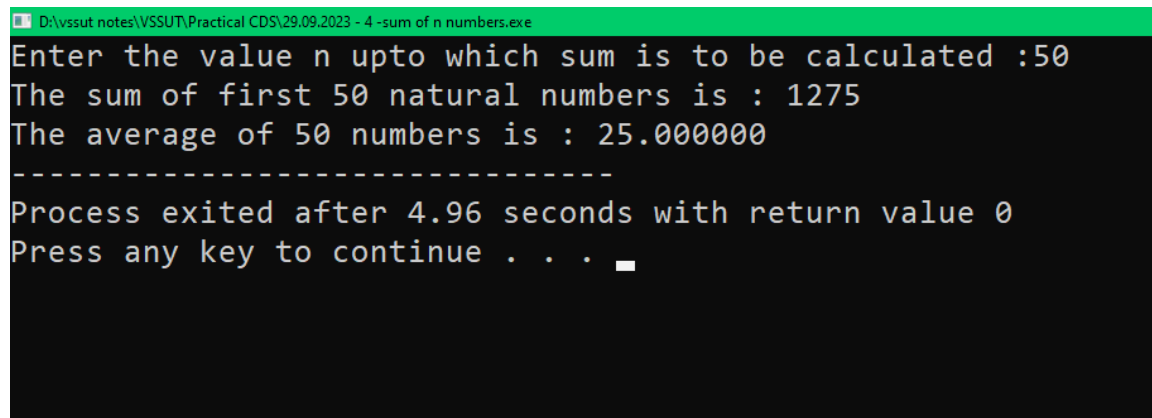
SOLUTIONS

By Samuel Priyatam

Q1. C program to find average and sum of n integers

```
#include <stdio.h>
int main(){
    int n,sum=0;
    printf("Enter the value n upto which sum is to be calculated :");
    scanf("%d",&n);

    for (int i=1;i<=n;i++){
        sum=sum+i;
    }
    printf("The sum of first %d natural numbers is : %d",n,sum);
    float avg= (float)(sum/n);
    printf("The average of %d numbers is : %f",n,avg);
    return 0;
}
```



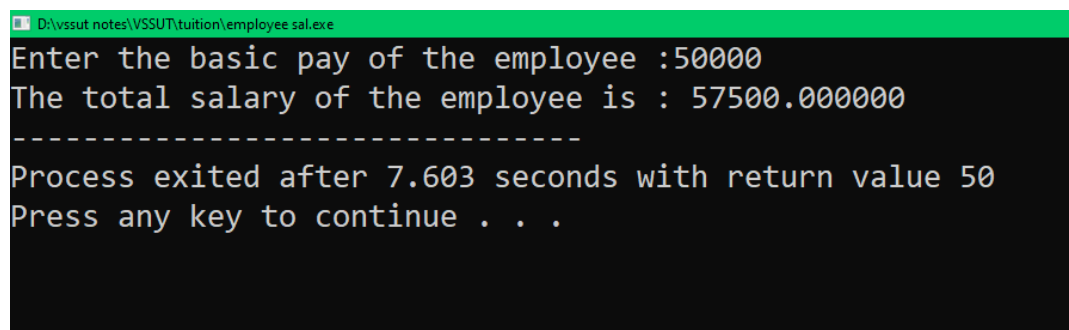
The screenshot shows a Windows command prompt window with a green title bar. The title bar text is "D:\vssut notes\VSSUT\Practical CDS\29.09.2023 - 4 -sum of n numbers.exe". The command prompt displays the following text:

```
Enter the value n upto which sum is to be calculated :50
The sum of first 50 natural numbers is : 1275
The average of 50 numbers is : 25.000000
-----
Process exited after 4.96 seconds with return value 0
Press any key to continue . . .
```

Q2. Write a program to calculate salary of an employee given his basic pay (to be entered by the user), HRA = 10% of the basic pay, TA = 5% of basic pay. Define HRA and TA as constants and use them to calculate the salary of the employee?

```
#include <stdio.h>
#define hra 0.1
#define ta 0.05

void main(){
    int bp;
    float sal,hrap,tap;
    printf("Enter the basic pay of the employee :");
    scanf("%d",&bp);
    hrap=hra*bp;
    tap=ta*bp;
    sal = bp+hrap+tap;
    printf("The total salary of the employee is : %f",sal);
}
```



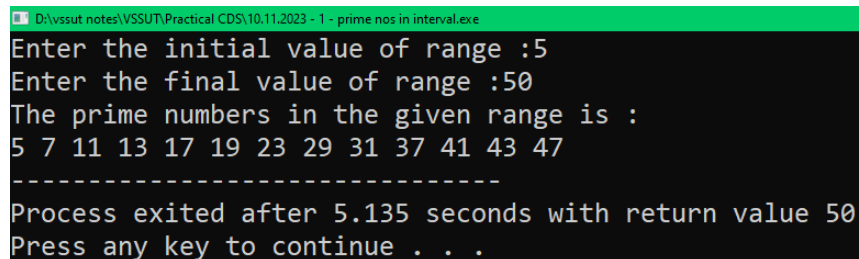
```
D:\vssut notes\VSSUT\tuition\employee sal.exe
Enter the basic pay of the employee :50000
The total salary of the employee is : 57500.000000
-----
Process exited after 7.603 seconds with return value 50
Press any key to continue . . .
```

Q3. C Programs to display prime numbers within a range

```
#include <stdio.h>

void primerange(){
    int a,b,i,j,c,temp;
    printf("Enter the initial value of range :");
    scanf("%d",&a);
    printf("Enter the final value of range :");
    scanf("%d",&b);
    if (a>b){
        temp=b;b=a;a=temp;}
    printf("The prime numbers in the given range is : \n");
    for (i=a;i<=b;i++){
        c=0;
        if (i<2)
            continue;
        for (j=2;j<i;j++){
            if(i%j==0){
                c=1;break;}
        }
        ((c==0)?printf("%d ",i):NULL);
    }
}

void main(){
    primerange();
}
```



The screenshot shows a Windows command prompt window with a green title bar. The title bar text is "D:\vssut notes\VSSUT\Practical CDS\10.11.2023 - 1 - prime nos in interval.exe". The command prompt displays the following text:

```
Enter the initial value of range :5
Enter the final value of range :50
The prime numbers in the given range is :
5 7 11 13 17 19 23 29 31 37 41 43 47
-----
Process exited after 5.135 seconds with return value 50
Press any key to continue . . .
```

Q4. WAP to enter 2 matrix and find their sum

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

```
void main(){
```

```
    int n,m,i,j;
```

```
    printf("Enter number of rows in matrix :");
```

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

```
    printf("Enter number of columns in matrix :");
```

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

```
    int a[n][m],b[n][m],c[n][m]; //matrix declaration
```

```
    printf("Enter the elements of the 1st matrix :\n");
```

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

```
        printf("%dth row\n",i+1);
```

```
        for(j=0;j<m;j++){
```

```
            printf("%d th element : ",j+1);
```

```
            scanf("%d",&a[i][j]);}
```

```
    }
```

```
    printf("Enter the elements of the 2nd matrix :\n");
```

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

```
        printf("%dth row\n",i+1);
```

```
        for(j=0;j<m;j++){
```

```
            printf("%d th element : ",j+1);
```

```
            scanf("%d",&b[i][j]);}
```

```
    }
```

```
    printf("Matrix A\n");
```

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

```
        for(j=0;j<m;j++){
```

```
            printf("%d ",a[i][j]);
```

```
        }
```

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

```
    printf("\nMatrix B\n");
```

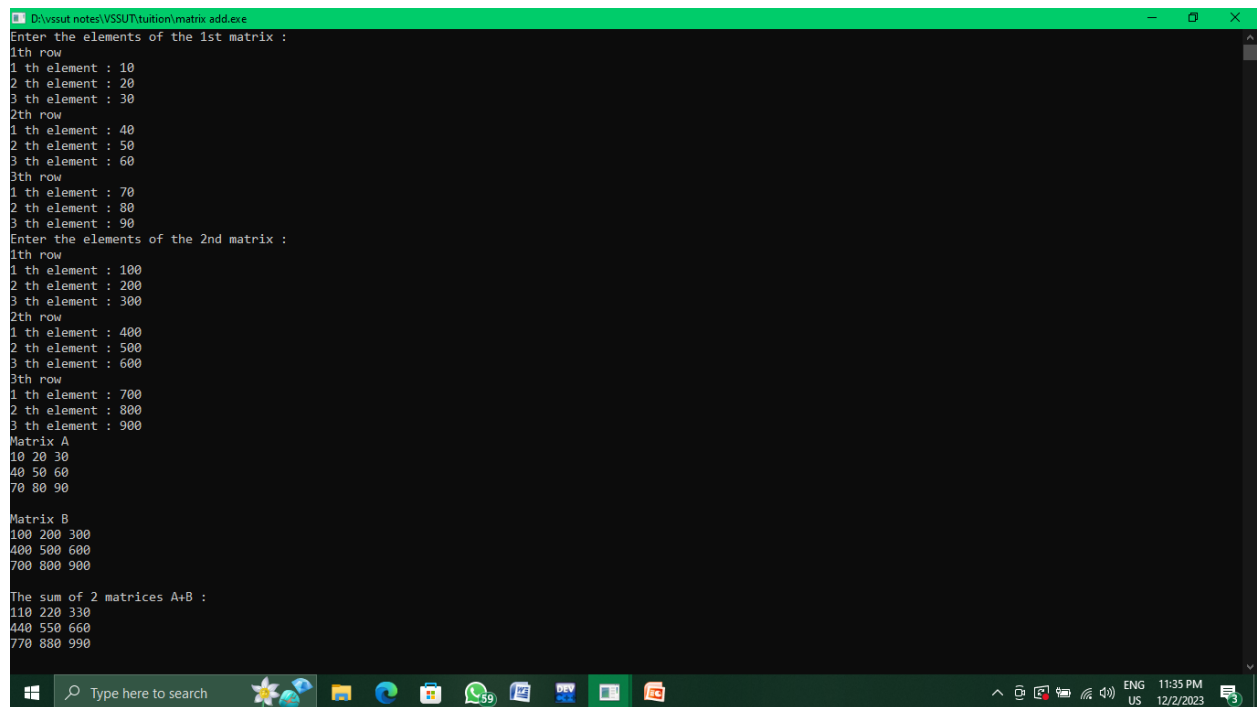
```
    for (i=0;i<n;i++){
```

```
        for(j=0;j<m;j++){
```

```

        printf("%d ",b[i][j]);
    }
    printf("\n");}
printf("\nThe sum of 2 matrices A+B :\n");
for (i=0;i<n;i++){
    for(j=0;j<m;j++){
        c[i][j]=a[i][j]+b[i][j];
        printf("%d ",c[i][j]);
    }
    printf("\n");
}
}

```



```

D:\vssut notes\VSSUT\tuition\matrix add.exe
Enter the elements of the 1st matrix :
1th row
1 th element : 10
2 th element : 20
3 th element : 30
2th row
1 th element : 40
2 th element : 50
3 th element : 60
3th row
1 th element : 70
2 th element : 80
3 th element : 90
Enter the elements of the 2nd matrix :
1th row
1 th element : 100
2 th element : 200
3 th element : 300
2th row
1 th element : 400
2 th element : 500
3 th element : 600
3th row
1 th element : 700
2 th element : 800
3 th element : 900
Matrix A
10 20 30
40 50 60
70 80 90
Matrix B
100 200 300
400 500 600
700 800 900
The sum of 2 matrices A+B :
110 220 330
440 550 660
770 880 990

```

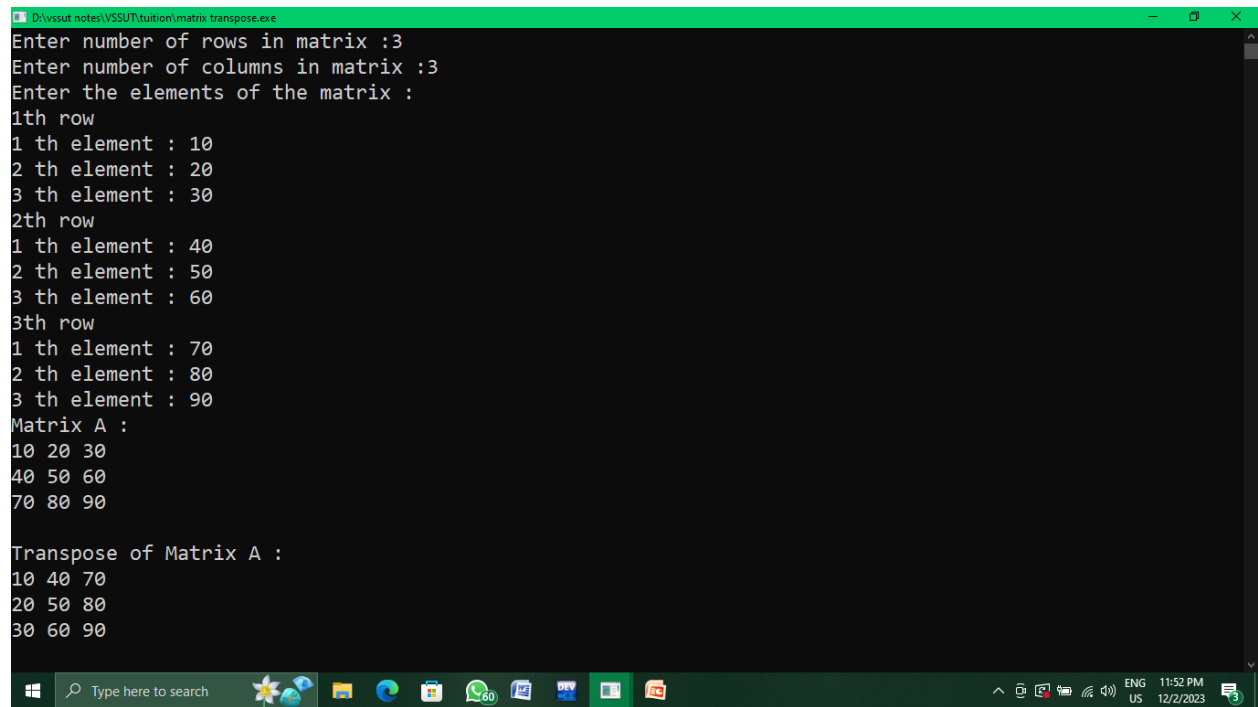
Q5. wap to find transpose of a matrix

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

```
void main(){
    int n,m,i,j;
    printf("Enter number of rows in matrix :");
    scanf("%d",&n);
    printf("Enter number of columns in matrix :");
    scanf("%d",&m);
    int a[n][m],b[m][n]; //matrix declaration
    printf("Enter the elements of the matrix :\n");
    for (i=0;i<n;i++){
        printf("%dth row\n",i+1);
        for(j=0;j<m;j++){
            printf("%d th element : ",j+1);
            scanf("%d",&a[i][j]);}
    }
    printf("Matrix A :\n");
    for (i=0;i<n;i++){
        for(j=0;j<m;j++){
            printf("%d ",a[i][j]);
        }
        printf("\n");
    }

    for (i=0;i<n;i++){
        for(j=0;j<m;j++){
            b[j][i]=a[i][j];
        }
    }
    printf("\nTranspose of Matrix A :\n");
```

```
        for (i=0;i<m;i++){  
            for(j=0;j<n;j++){  
                printf("%d ",b[i][j]);  
            }  
            printf("\n");  
        }  
    }  
}
```

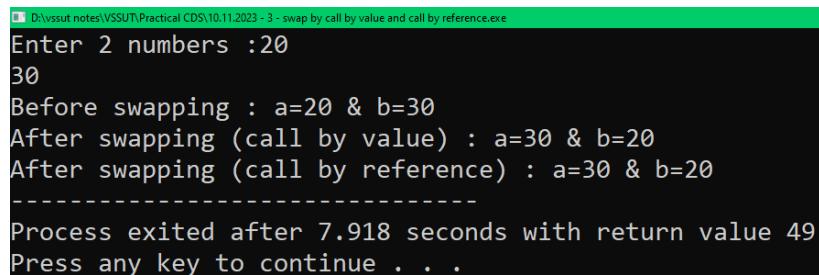


The screenshot shows a Windows command prompt window titled "D:\vssut notes\VSSUT\tuition\matrix transpose.exe". The program prompts the user to enter the number of rows (3), the number of columns (3), and the elements of the matrix. It then displays the original matrix A and its transpose.

```
D:\vssut notes\VSSUT\tuition\matrix transpose.exe  
Enter number of rows in matrix :3  
Enter number of columns in matrix :3  
Enter the elements of the matrix :  
1th row  
1 th element : 10  
2 th element : 20  
3 th element : 30  
2th row  
1 th element : 40  
2 th element : 50  
3 th element : 60  
3th row  
1 th element : 70  
2 th element : 80  
3 th element : 90  
Matrix A :  
10 20 30  
40 50 60  
70 80 90  
  
Transpose of Matrix A :  
10 40 70  
20 50 80  
30 60 90
```

Q6. Swapping of 2 numbers using call by value and call by reference method

```
#include <stdio.h>
int a,b;
void swapint(int c,int d){
    int temp=c;
    c=d;d=temp;
    printf("\nAfter swapping (call by value) : a=%d & b=%d",c,d);
}
void swappointer(int *p,int *q){
    int temp;
    temp=*p; *p=*q; *q=temp;
}
void main(){
    printf("Enter 2 numbers :");
    scanf("%d%d",&a,&b);
    printf("Before swapping : a=%d & b=%d",a,b);
    swapint(a,b);
    swappointer(&a,&b);
    //a=5
    printf("\nAfter swapping (call by reference) : a=%d & b=%d", a,b);
}
```



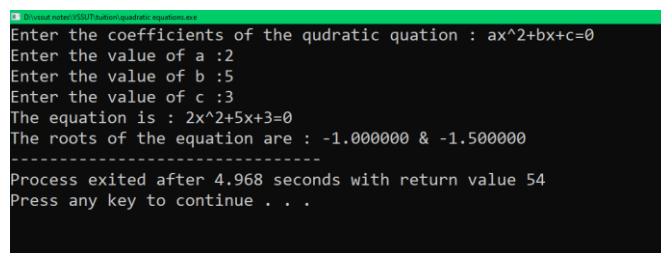
D:\vssut notes\VSSUT\Practical CDS\10.11.2023 - 3 - swap by call by value and call by reference.exe

```
Enter 2 numbers :20
30
Before swapping : a=20 & b=30
After swapping (call by value) : a=30 & b=20
After swapping (call by reference) : a=30 & b=20
-----
Process exited after 7.918 seconds with return value 49
Press any key to continue . . .
```

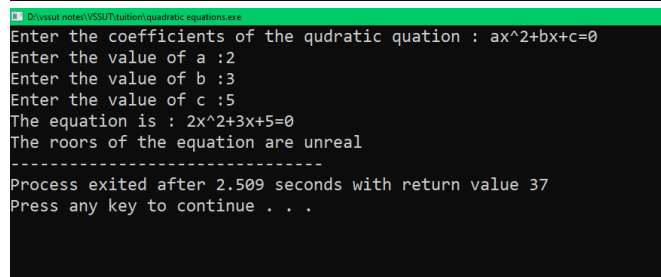

Q7. Find out all the roots of a quadratic equation by the following expression

$$ax^2+bx+c = 0$$

```
#include <stdio.h>
#include <math.h>
void main(){
    int a,b,c;
    printf("Enter the coefficients of the quadratic equation : ax^2+bx+c=0 \n");
    printf("Enter the value of a :");
    scanf("%d",&a);
    printf("Enter the value of b :");
    scanf("%d",&b);
    printf("Enter the value of c :");
    scanf("%d",&c);
    printf("The equation is : %dx^2+%dx+%d=0",a,b,c);
    if ((pow(b,2)-4*a*c)<0)
        printf("\nThe roots of the equation are unreal");
    else{
        float p=(float)(-b+sqrt(pow(b,2)-4*a*c))/(2*a);
        float q=(float)(-b-sqrt(pow(b,2)-4*a*c))/(2*a);
        printf("\nThe roots of the equation are : %f & %f",p,q);
    }
}
```



```
D:\vsut notes\VSUT\quadratic equations.exe
Enter the coefficients of the quadratic equation : ax^2+bx+c=0
Enter the value of a :2
Enter the value of b :5
Enter the value of c :3
The equation is : 2x^2+5x+3=0
The roots of the equation are : -1.000000 & -1.500000
-----
Process exited after 4.968 seconds with return value 54
Press any key to continue . . .
```

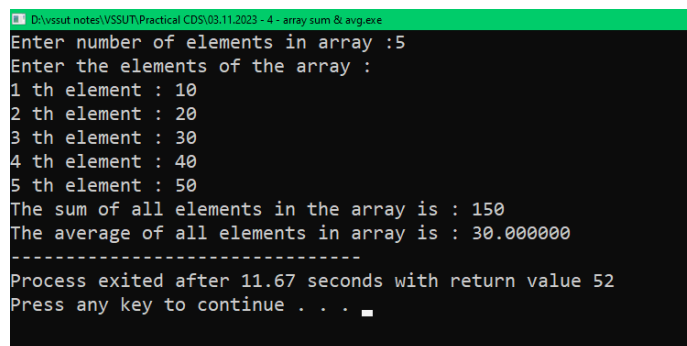


```
D:\vsut notes\VSUT\quadratic equations.exe
Enter the coefficients of the quadratic equation : ax^2+bx+c=0
Enter the value of a :2
Enter the value of b :3
Enter the value of c :5
The equation is : 2x^2+3x+5=0
The roots of the equation are unreal
-----
Process exited after 2.509 seconds with return value 37
Press any key to continue . . .
```

Q8. WAP to create an array and find sum & average of element present in array

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

```
void main(){
    int n,i,s=0;
    printf("Enter number of elements in array :");
    scanf("%d",&n);
    int a[n];
    printf("Enter the elements of the array :\n");
    for (i=0;i<n;i++){
        printf("%d th element : ",i+1);
        scanf("%d",&a[i]);
    }
    for (i=0;i<n;){
        s+=a[i++];
    }
    printf("The sum of all elements in the array is : %d",s);
    printf("\nThe average of all elements in array is : %f",(float)s/n);
}
```



```
D:\vssut notes\VSSUT\Practical CDS\03.11.2023 - 4 - array sum & avg.exe
Enter number of elements in array :5
Enter the elements of the array :
1 th element : 10
2 th element : 20
3 th element : 30
4 th element : 40
5 th element : 50
The sum of all elements in the array is : 150
The average of all elements in array is : 30.000000
-----
Process exited after 11.67 seconds with return value 52
Press any key to continue . . .
```

Theory Questions

Q1. Explain the steps involved in compilation and execution of a C program with suitable flow diagram

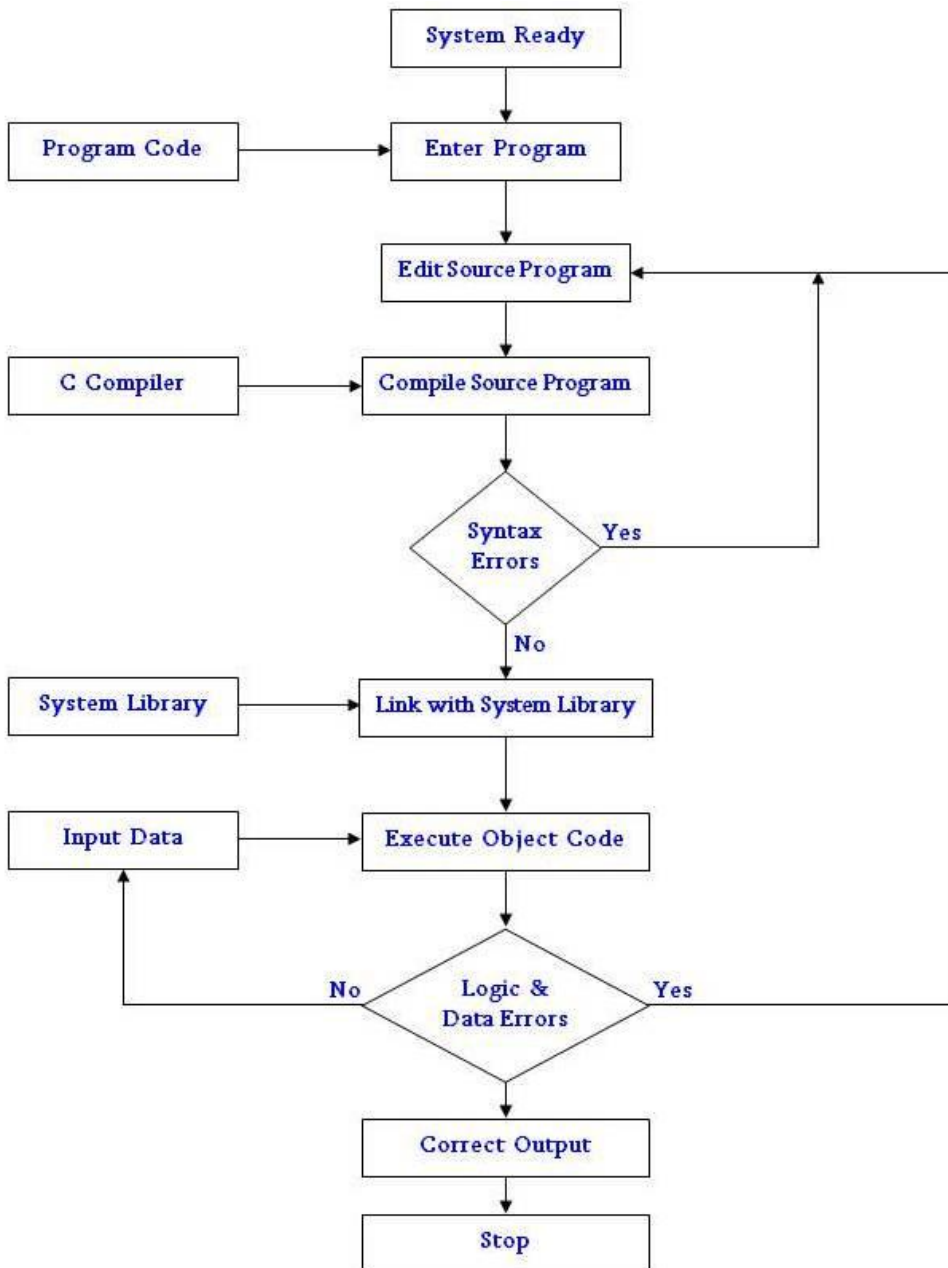


Figure : Process of compiling and running a C program.

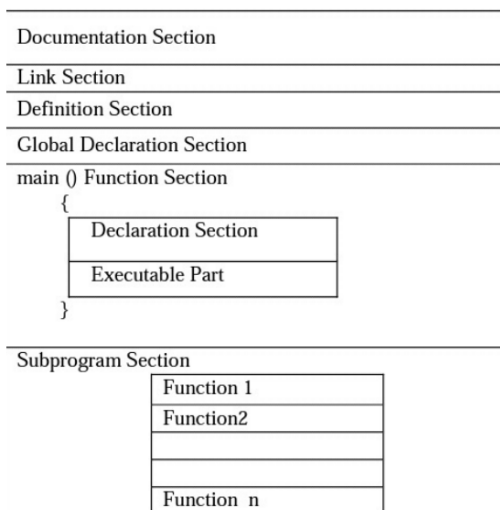
Q2. Describe the C structure

Structure of a C Program

A C program is divided into different sections.

There are six main sections in a basic c program.

The six sections are: 1. Documentation Section 2. Link Section 3. Definition Section 4. Global Declarations Section 5. Main functions Section 6. Subprograms Section



Q3. How is a compiler different from an interpreter?

- [A compiler translates the entire source code into machine code before execution, whereas an interpreter translates code line by line during execution¹².](#)
- [A compiler generates an object code that can be saved and reused, whereas an interpreter does not produce any intermediate code³.](#)
- A compiler usually takes more time to analyze the source code, but the overall execution time is faster. [An interpreter usually takes less time to analyze the source code, but the overall execution time is slower⁴](#)
- A compiler is more suitable for languages that require high performance and optimization, such as C, C++, and Java. [An interpreter is more suitable for languages that require dynamic features and portability, such as Python, Ruby, and JavaScript⁵](#)

Q4. Differentiate between algorithm and flowchart.

Difference between Algorithm and Flowchart	
Algorithm	Flowchart
An algorithm is a step by step process to solve a problem.	A flowchart is graphical representation of algorithm.
Difficult to show branching and looping	Easy to show branching and looping.
Algorithm can be written for any problem	Flowchart for big problem is impractical
Easy to debug errors	Difficult to debug errors.

Q5. Write algorithms and flowchart for various problems:

- i- Write algorithm and draw flowchart to find largest among 3 numbers

write algorithm & flowchart to find largest among 3 numbers.

Algorithm step 0 - start

step 1 - Read / input 3 numbers a, b, c

step 2 - Compare $a > b$ AND $a > c$

~~step 2~~ if TRUE, greater = a

if FALSE,

Compare $b > a$ AND $b > c$

if TRUE, greater = b

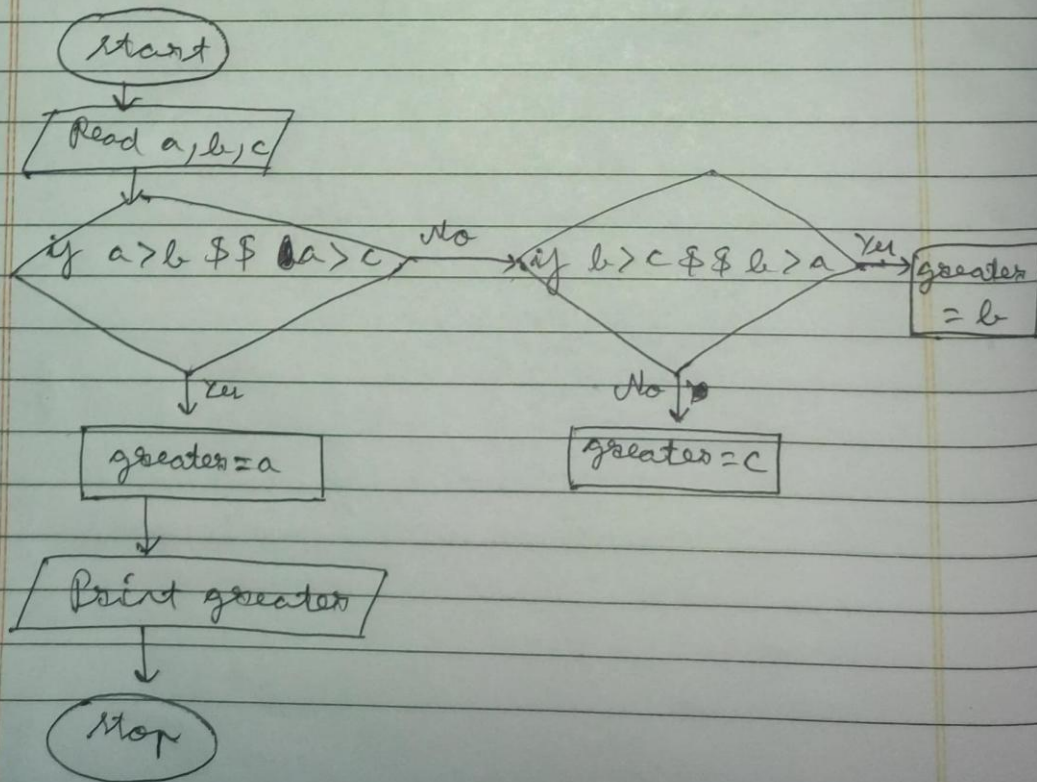
else

greater = c

step 3 - Display greater

step 4 - stop

Flowchart :-



ii- Find out all the roots of a quadratic equation by the following expression $ax^2+bx+c=0$

- Write an algorithm for computing all the roots.
- Draw a flowchart for the above algorithm

Q- WAP to find roots of a quadratic equation & write its algorithm & flowchart.

A- Algorithm

Step 1 - Start

Step 2 - Read/input 3 numbers a, b, c

Step 3 - Display equation: $a*x^2 + b*x + c = 0$

Step 4 - Compare $(b^2 - 4*a*c) < 0$

if TRUE
Display "Roots are unreal"

else
evaluate
$$P = \frac{-b + \sqrt{b^2 - 4*a*c}}{2*a}$$
$$Q = \frac{-b - \sqrt{b^2 - 4*a*c}}{2*a}$$
Display "Roots are: P, Q"

Step 5 - Stop

Flowchart

Q6. Differentiate between syntax error and logical error.

A syntax error is an error that occurs when the code does not follow the rules of the programming language. For example, if you forget to put a semicolon at the end of a statement, or if you misspell a keyword, you will get a syntax error. A syntax error will prevent the code from compiling or running.

A logical error is an error that occurs when the code does not do what you intended it to do. For example, if you use the wrong operator, or if you make a mistake in the algorithm, you will get a logical error. A logical error will not stop the code from running, but it will produce incorrect results.

Here is an example of a syntax error and a logical error in C++:

```
#include <iostream>
using namespace std;
int main()
{
    int x = 10;
    int y = 5;
    // Syntax error: missing semicolon
    cout << "x + y = " << x + y
    // Logical error: wrong operator
    cout << "x - y = " << x * y << endl;
    return 0;
}
```

Q7. Differentiate between while and do-while loop

- While Loop

- i- A while loop is the most straightforward looping structure
- ii- It is entry controlled loop.
- iii- The basic format of while loop is as follows:
 while (condition) {
 statements;
 }

- Do-While loop

- i- A do-while loop is similar to the while loop except that the condition is always executed after the body of a loop
- ii- It is also called an exit controlled loop.
- iii- The basic format of while loop is as follows:
 do
 {
 Statements
 }
 while (expression);

do...while loop guaranteed to execute at least one time whereas while loop does not

Q8. Differentiate between break and continue statement.

i- break statement

The break statement provides an early exit from loops like for, while, and do..while.

A break causes the innermost enclosing loop or switch to be exited immediately.

When break is encountered inside any loop, control automatically passes to the first statement after the loop.

ii- Continue statement

It is sometimes desirable to skip some statements inside the loop.

In such cases, continue statement is used. The continue statement is related to break, but less often used; it causes the next iteration of the enclosing for, while, or do loop to begin.

In the while and do, this means that the test part is executed immediately; in the for, control passes to the increment step.

The continue statement applies only to loops, not to switch.

Q9. What is operator precedence and associativity?

Operator precedence and associativity are two concepts that determine how an expression with multiple operators is evaluated. Operator precedence refers to the order of priority given to different operators, while operator associativity refers to the direction in which operators with the same precedence are executed.

For example, in the expression $2 + 3 * 4$, the operator $*$ has higher precedence than $+$, so it is evaluated first. The result is $2 + 12$, which is then evaluated to 14. This is equivalent to $(2 + (3 * 4))$.

However, in the expression $2 * 3 / 4$, the operators $*$ and $/$ have the same precedence, so they are evaluated according to their associativity. In this case, the associativity is left-to-right,

so the expression is evaluated as $(2 * 3) / 4$, which is $6 / 4$, which is 1.5

Q10.

```
#include <stdio.h>
int main() {
    do
        printf("In while loop ");
    while (0);
    printf("After loop\n");
}
```

Output :

In while loop

Q11.

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

```
int main()
```

```
{  
    while ()  
        printf("In while loop ");  
    printf("After loop\n");  
}
```

Output :

Expression Syntax Error

Q12.

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

```
int main ( )
```

```
{  
    int a=6,b=4;  
    while (a+b)  
    {  
        printf( "a=%d, b=%d\n", a, b) ;  
        a=a/2;  
        b%=3;  
    }  
    return 0;  
}
```

output :

a=6, b=4

a=3, b=1

a=1, b=1

a=0, b=1

a=0, b=1

.

.

.

.

(infinite loop)

Q13.

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

```
int main ( )
```

```
{
```

```
    int m=10,n=6;
```

```
    while (m+n)
```

```
    {
```

```
        printf( "m=%d, n=%d\n", m, n) ;
```

```
        m=m/2;
```

```
        n%=3;
```

```
    }
```

```
    return 0;
```

```
}
```

Output :

m=10, n=6

m=5, n=0

m=2, n=0

m=1, n=0