AIM

1. Implement a C program to display Fibonacci series using recursion.
2. Implement a C program to display factorial of a number using recursion.
3. Implement a C program to display array in reverse and calculate the items in an array using recursion.
4. Implement a C program to assign members to a structure and display it.(Use example of your choice).
5. Display shopping list using structures accessed with its pointer.

THEORY

RECURSIVE FUCNTION: A function that calls itself is called a recursive function.

Recursive call always leads to an infinite loop. So, provision must be mase to get outside this infinite loop.

Recursion are mainly of two types depending on weather a function calls itself from within itself weather two function call one another mutually. The former is called direct recursion and t latter is called indirect recursion. Thus, the two types of recursion are:

1. Direct recursion
2. Indirect recursion

DIRECT RECURSION	INDIRECT RECURSION
In the direct recursion, only one function is called by itself.	In indirect recursion more than one function are by the other function and number of times.

Int num()
 	
{
 	
Num()
 	
}
	Int num(){
	
	Sum()
	
	}
	
	Int sum()
	
	{
	
	Num()
	
	}

Linear recursion

It is the most commonly used recursion, where a function calls itself in simple manner and a terminating condition is used to terminate the recursion. Forwarding recursion is called winding and getting the control back to the caller is called unwinding.

Tail recursion

It is the recursion where recursive function is called at the end of recursive function.

Mutual recursion:

Calling two or more functions mutual is called mutual recursion. Say for example, if function A is calling B and function B is calling A recursively then it is said that, they are in mutual recursion.

Nested recursion:

When a recursive method has a parameter defined in terms of itself then it is called nested recursion

STRUCTURE is a collection of dissimilar elements(usually) stored in adjacent locations. They are also known as User-Defined data types.

Syntax

```
struct structure_name

{

        int a;

        char b;

        float c;

} e1, e2;
```

Where struct is a keyword,

a, b, c are the structure elements

e1, e2 are the structure variables

Uses of Structures

1) Database Management
2) Interaction with Mouse, etc.

To access structure elements using structure pointer, use -> operator.

Struct emp e;

Struct emp *p;

p=&e;

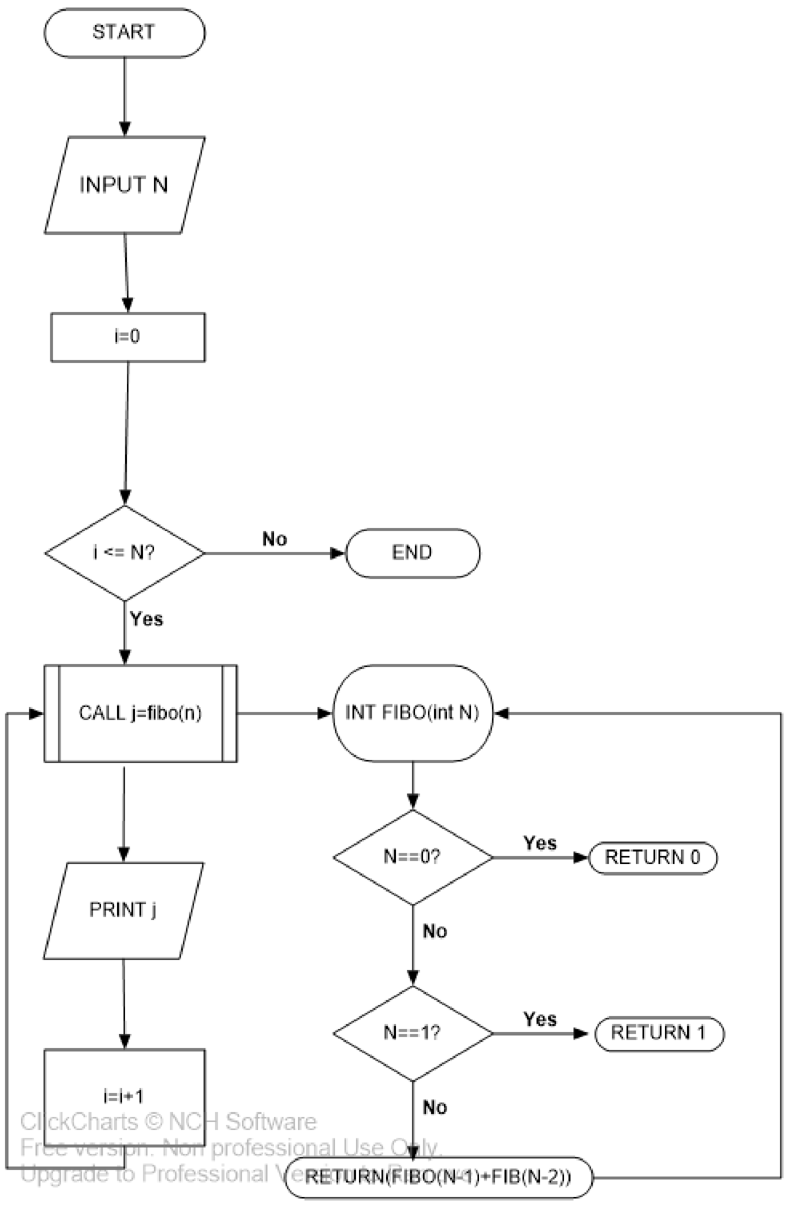
printf("%%s %d %f",p->name,p->age,p->salary);

PSEUDO CODE

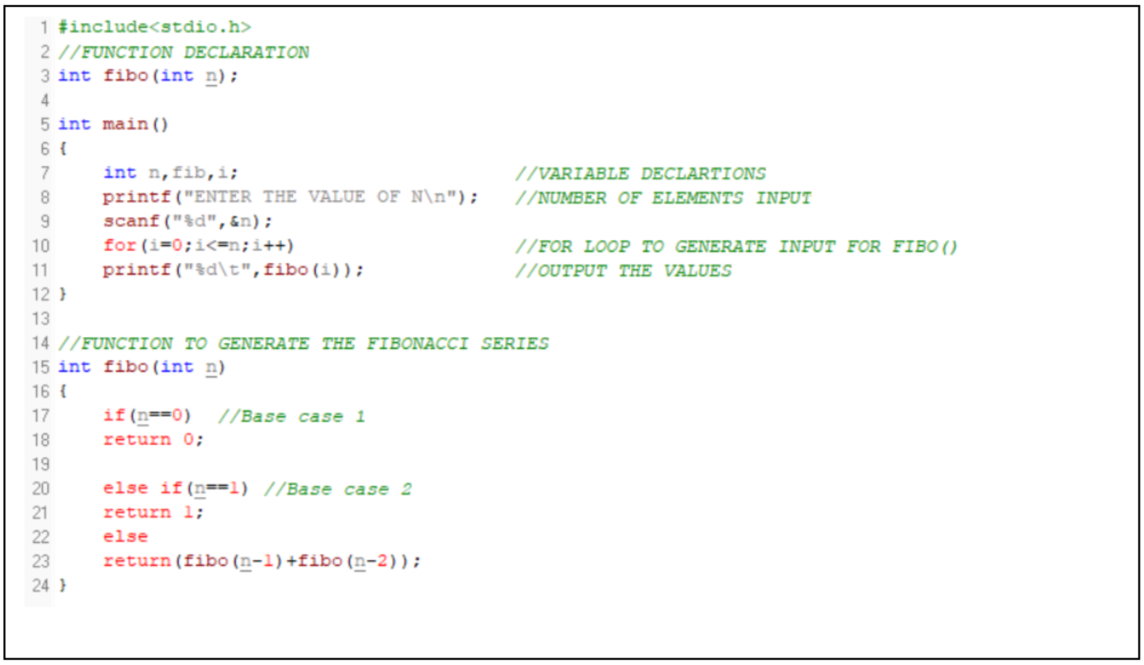
1)

```
1. START
2. INPUT N
3. FOR i=0…..N
    1. J= FIBO(N)
    2. PRINT J
4. END


INT FIBO(INT N)

1. IF N==0
    1. RETURN(0)

2. ELSE if N==1

    1.RETURN(1)

3. ELSE

    1. RETURN(FIBO(N-1)+FIBO(N-2))
```

FLOW CHART

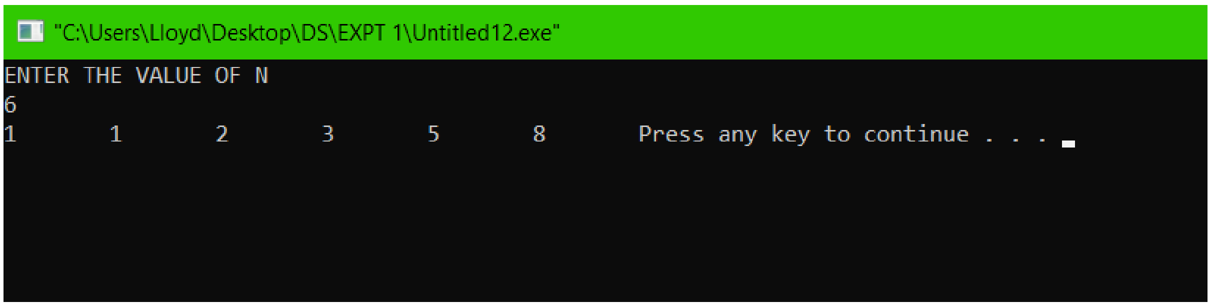

START
INPUT N
i=0
i <= N?
No
END
Yes
CALL j=fibo(n)
INT FIBO(int N)
N==0?
Yes
RETURN 0
No
PRINT j
N==1?
Yes
RETURN 1
No
i=i+1
RETURN(FIBO(N-1)+FIB(N-2))

SOURCE CODE



```c
#include<stdio.h>
//FUNCTION DECLARATION
int fibo(int n);

int main()
{
    int n,fib,i;                        //VARIABLE DECLARTIONS
    printf("ENTER THE VALUE OF N\n");   //NUMBER OF ELEMENTS INPUT
    scanf("%d",&n);
    for(i=0;i<=n;i++)                   //FOR LOOP TO GENERATE INPUT FOR FIBO()
    printf("%d\t",fibo(i));             //OUTPUT THE VALUES
}

//FUNCTION TO GENERATE THE FIBONACCI SERIES
int fibo(int n)
{
    if(n==0)  //Base case 1
    return 0;

    else if(n==1) //Base case 2
    return 1;
    else
    return(fibo(n-1)+fibo(n-2));
}
```

OUTPUT



"C:\Users\Lloyd\Desktop\DS\EXPT 1\Untitled12.exe"

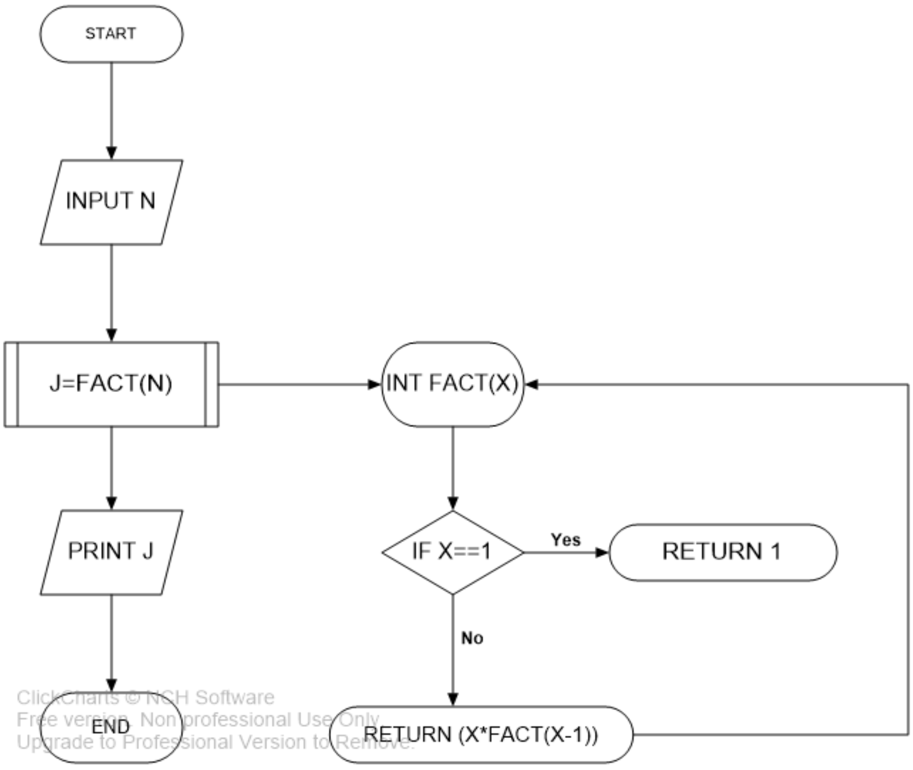
```
ENTER THE VALUE OF N
6
1       1       2       3       5       8       Press any key to continue . . .
```

2)

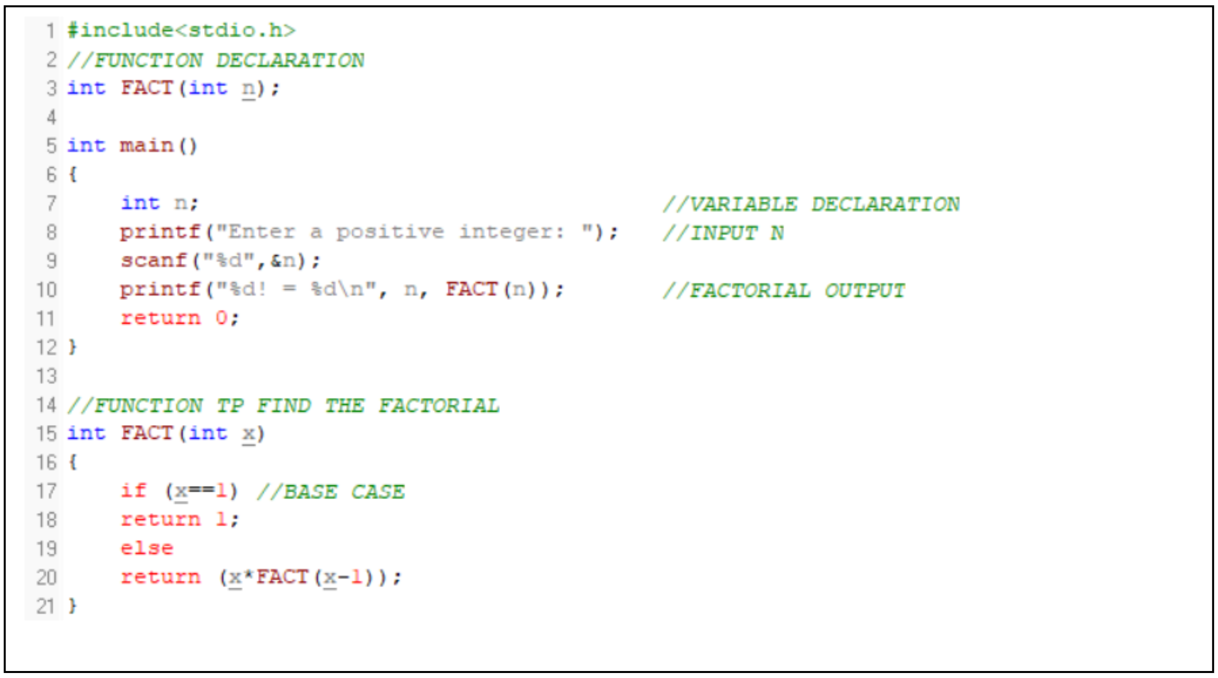
PSEUDO CODE

```
1. START
2. INPUT N
3. PRINT FACT(N)
4. END

INT FACT(X)
1. IF X==1
   1. RETURN 1
2. ELSE
   1. RETURN(X*FACT(X-1))
```

FLOWCHART

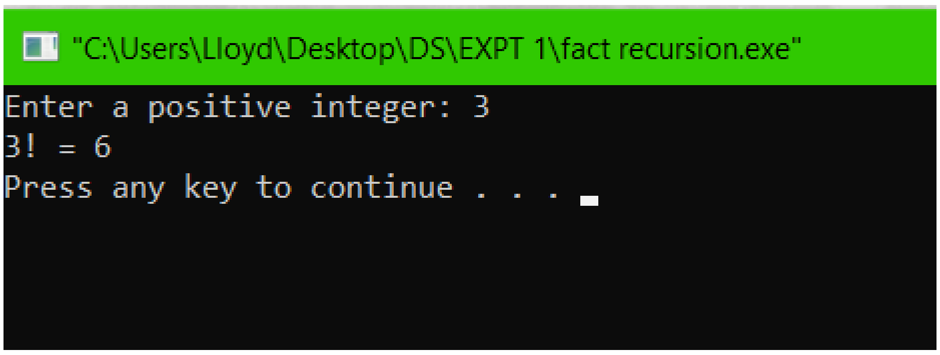


SOURCE CODE



```c
#include<stdio.h>
//FUNCTION DECLARATION
int FACT(int n);

int main()
{
    int n;                                  //VARIABLE DECLARATION
    printf("Enter a positive integer: ");   //INPUT N
    scanf("%d",&n);
    printf("%d! = %d\n", n, FACT(n));       //FACTORIAL OUTPUT
    return 0;
}

//FUNCTION TP FIND THE FACTORIAL
int FACT(int x)
{
    if (x==1) //BASE CASE
    return 1;
    else
    return (x*FACT(x-1));
}
```

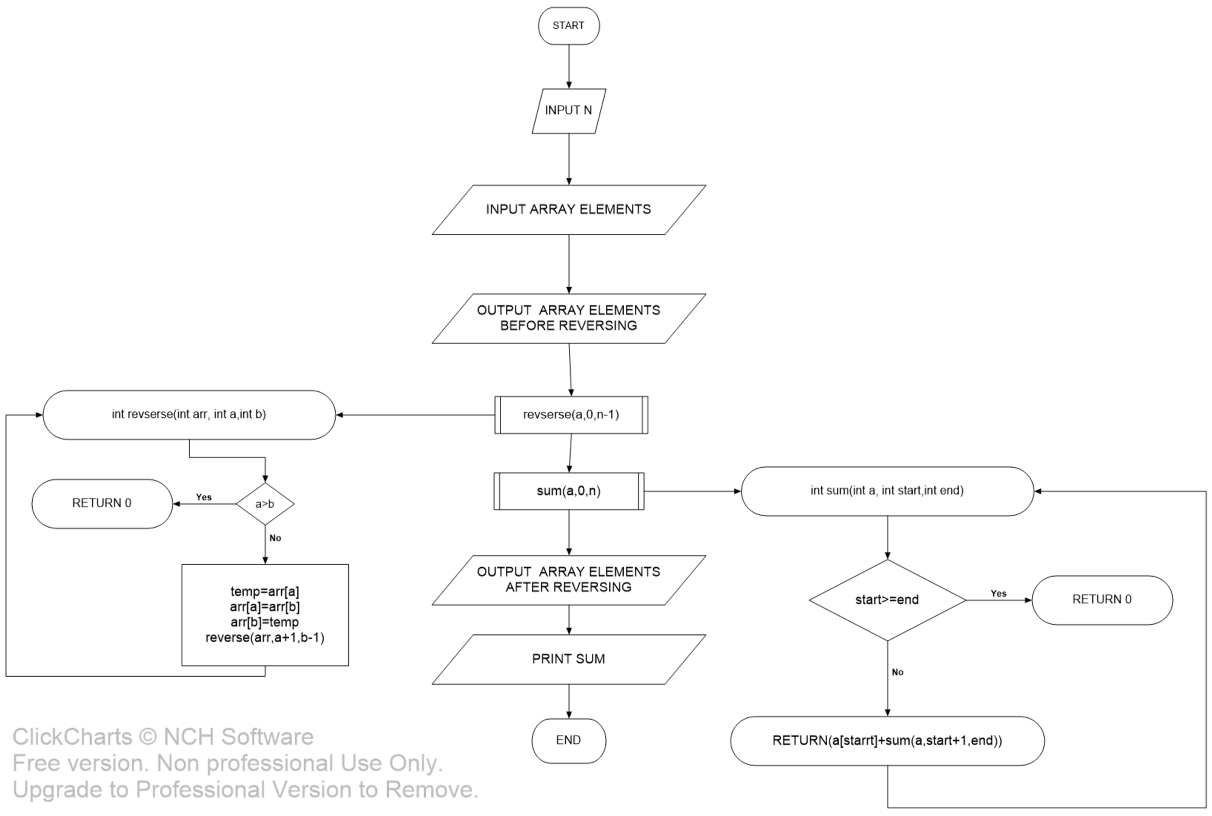
OUTPUT:



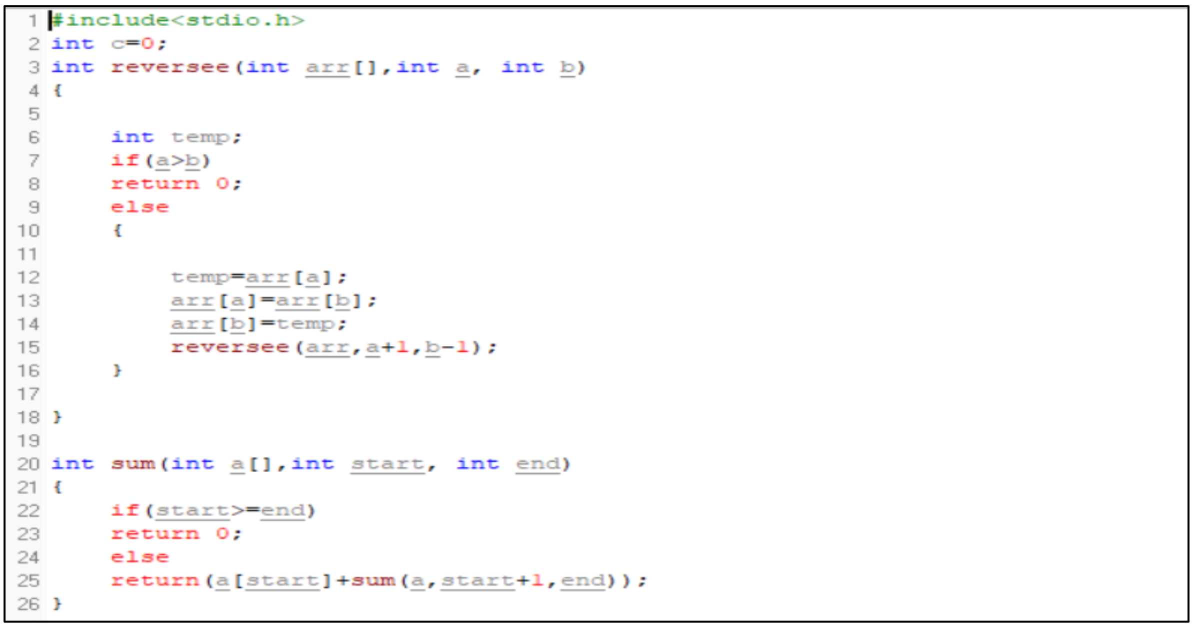
```
"C:\Users\Lloyd\Desktop\DS\EXPT 1\fact recursion.exe"
Enter a positive integer: 3
3! = 6
Press any key to continue . . .
```

3)

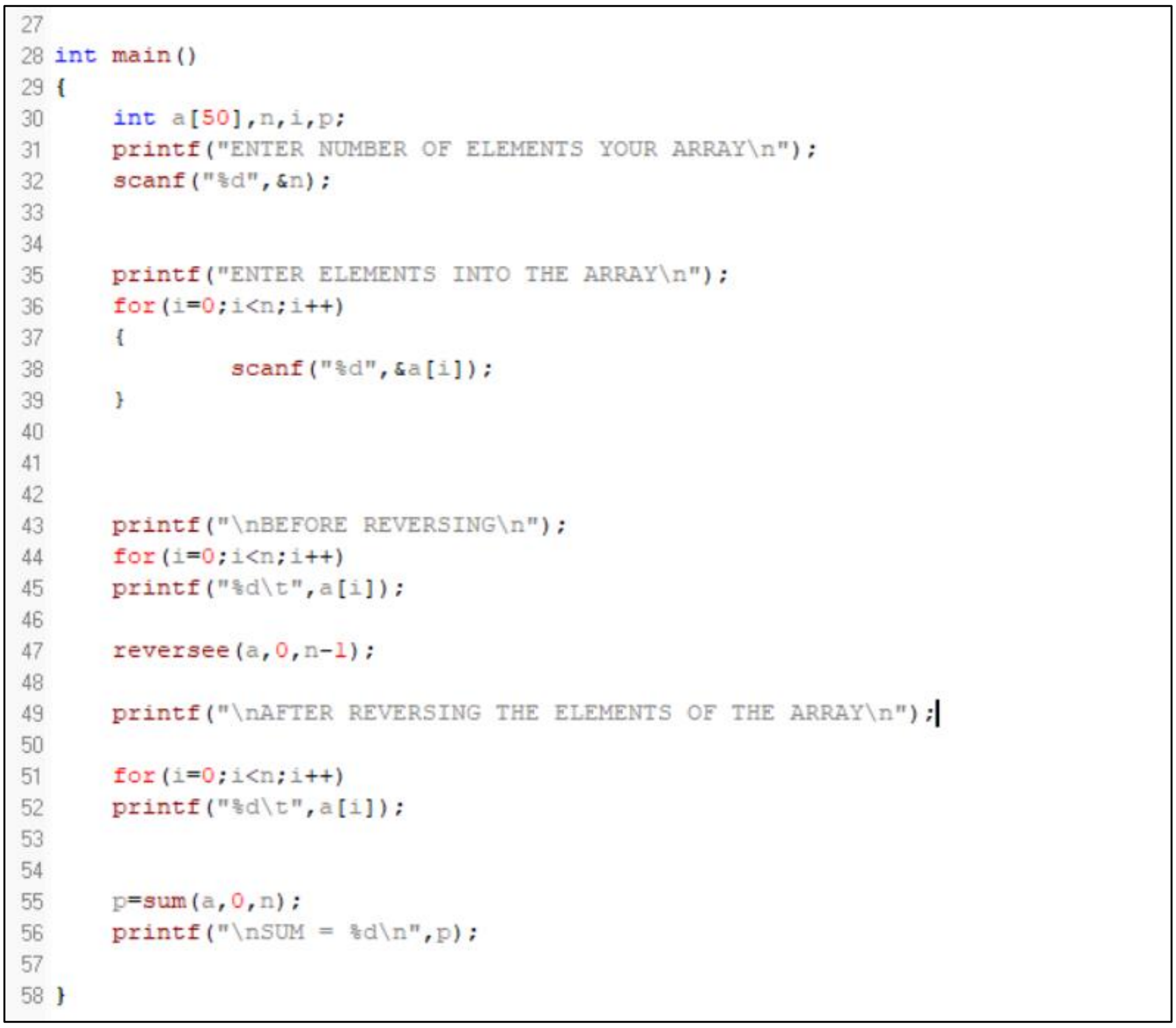
```
1. START
2. INPUT NUMBER OF ELEMENTS
3. INPUT ELEMEMTS INTO THE ARRAY
4. PRINT THE ARRAY BEOFRE REVERSING
5. CALL reverse(array,0,n-1)
6. PRINT THE ARRAY AFTER REVERSING
7. PRINT THE SUM

Void reverse(int arr[], int a, int b)
1. IF a>b
        1.RETURN 0;
2. ELSE
        1. temp=arr[a]
        2. arr[a]=arr[b]
        3. arr[b]=temp
        4. reverse(arr,a+1,b-1)

Void sum(int a[], int start, int end)
1. IF start>=end
        RETURN 0
2. ELSE
        RETURN(a[start]+sum(a,start+1,end)
```


START
INPUT N
INPUT ARRAY ELEMENTS
OUTPUT ARRAY ELEMENTS BEFORE REVERSING
revserse(a,0,n-1)
sum(a,0,n)
OUTPUT ARRAY ELEMENTS AFTER REVERSING
PRINT SUM
END
int revserse(int arr, int a,int b)
a>b
Yes
RETURN 0
No
temp=arr[a]
arr[a]=arr[b]
arr[b]=temp
reverse(arr,a+1,b-1)
int sum(int a, int start,int end)
start>=end
Yes
RETURN 0
No
RETURN(a[starrt]+sum(a,start+1,end))

SOURCE CODE



```c
#include<stdio.h>
int c=0;
int reversee(int arr[],int a, int b)
{

    int temp;
    if(a>b)
    return 0;
    else
    {

        temp=arr[a];
        arr[a]=arr[b];
        arr[b]=temp;
        reversee(arr,a+1,b-1);
    }

}

int sum(int a[],int start, int end)
{
    if(start>=end)
    return 0;
    else
    return(a[start]+sum(a,start+1,end));
}
```



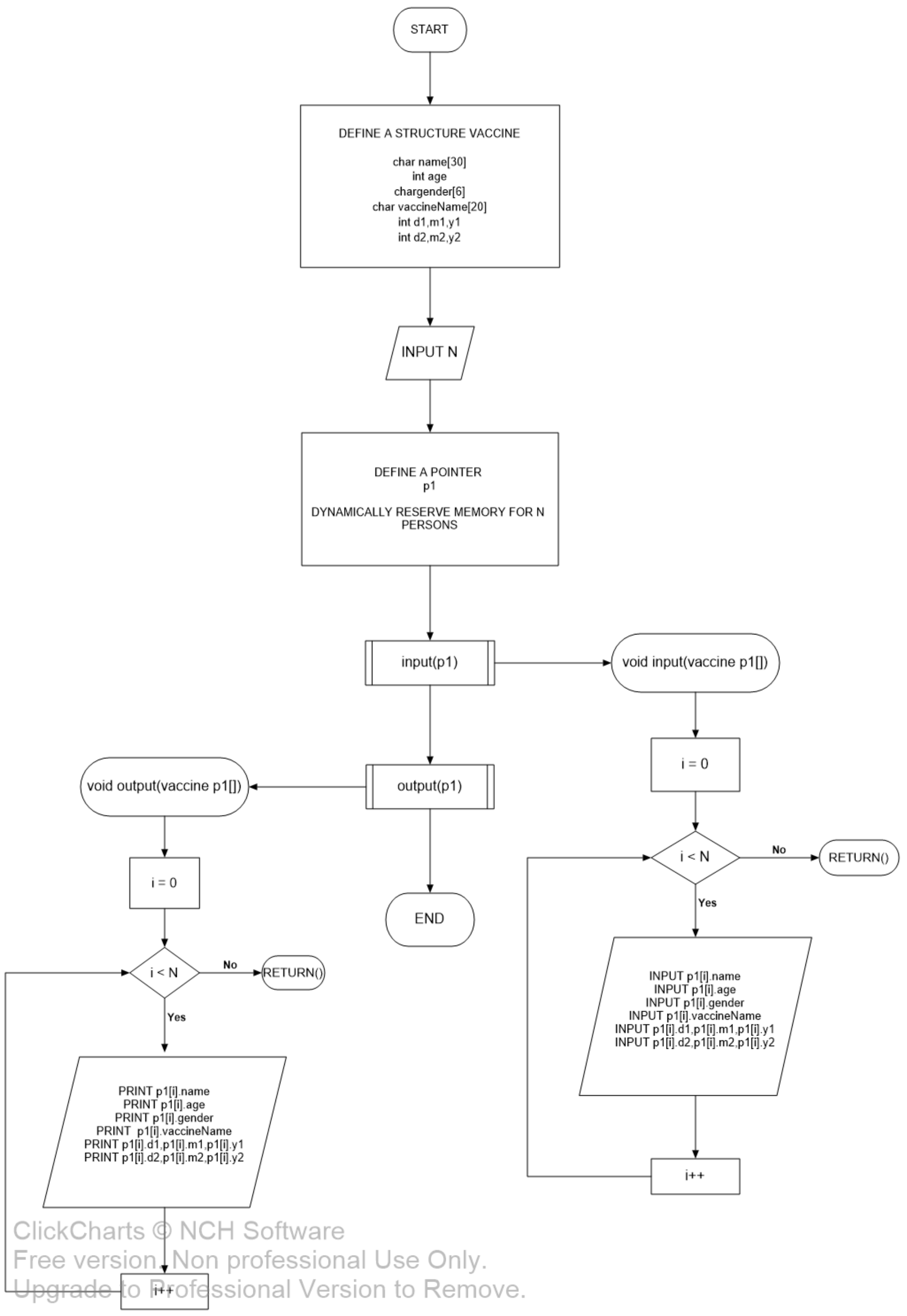
```c

int main()
{
    int a[50],n,i,p;
    printf("ENTER NUMBER OF ELEMENTS YOUR ARRAY\n");
    scanf("%d",&n);


    printf("ENTER ELEMENTS INTO THE ARRAY\n");
    for(i=0;i<n;i++)
    {
            scanf("%d",&a[i]);
    }



    printf("\nBEFORE REVERSING\n");
    for(i=0;i<n;i++)
    printf("%d\t",a[i]);

    reversee(a,0,n-1);

    printf("\nAFTER REVERSING THE ELEMENTS OF THE ARRAY\n");

    for(i=0;i<n;i++)
    printf("%d\t",a[i]);


    p=sum(a,0,n);
    printf("\nSUM = %d\n",p);

}
```

4)

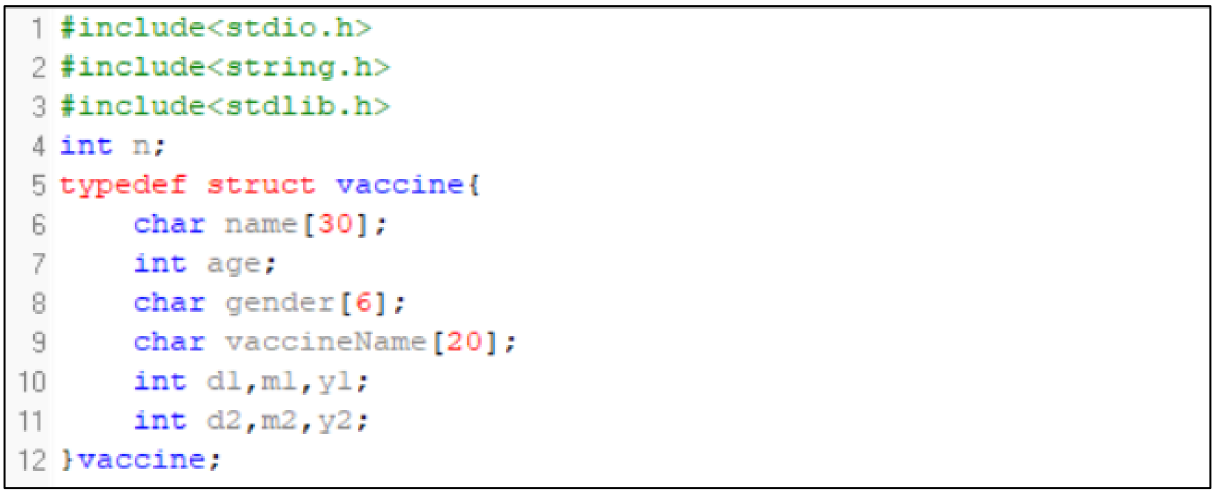
PSEUDO CODE

```
1. START
2.  DEFINE A STRUCTURE VACCIINE
        1. char name[30]
        2. int age
        3. char gender[6]
        4. char vaccineName[20]
        5. int d1,m1,y1
        6. int d2,m2,y2
3. INPUT N
4. DECLARE A POINTER p1 OF TYPE VACCINE AND DYNAMICALLY  RESERVE MEMORY FOR N PERSONS
5. CALL input(p1)
6. CALL output(p1)
7.END


void  output(vaccine p[])
1. i=0
2. FOR i < N
        1. INPUT p1[i].name, p1[i].age, p1[i].gender, p1[i].vaccineName ,p1[i].d1,p1[i].m1,p1[i].y1,
        p1[i].d2,p1[i].m2,p1[i].y2
        2. i++


void output(vaccine p[])
1. i=0
2. FOR i<N
        1. PRINT p1[i].name, p1[i].age, p1[i].gender, p1[i].vaccineName ,p1[i].d1,p1[i].m1,p1[i].y1,
        p1[i].d2,p1[i].m2,p1[i].y2
        2. i++
```

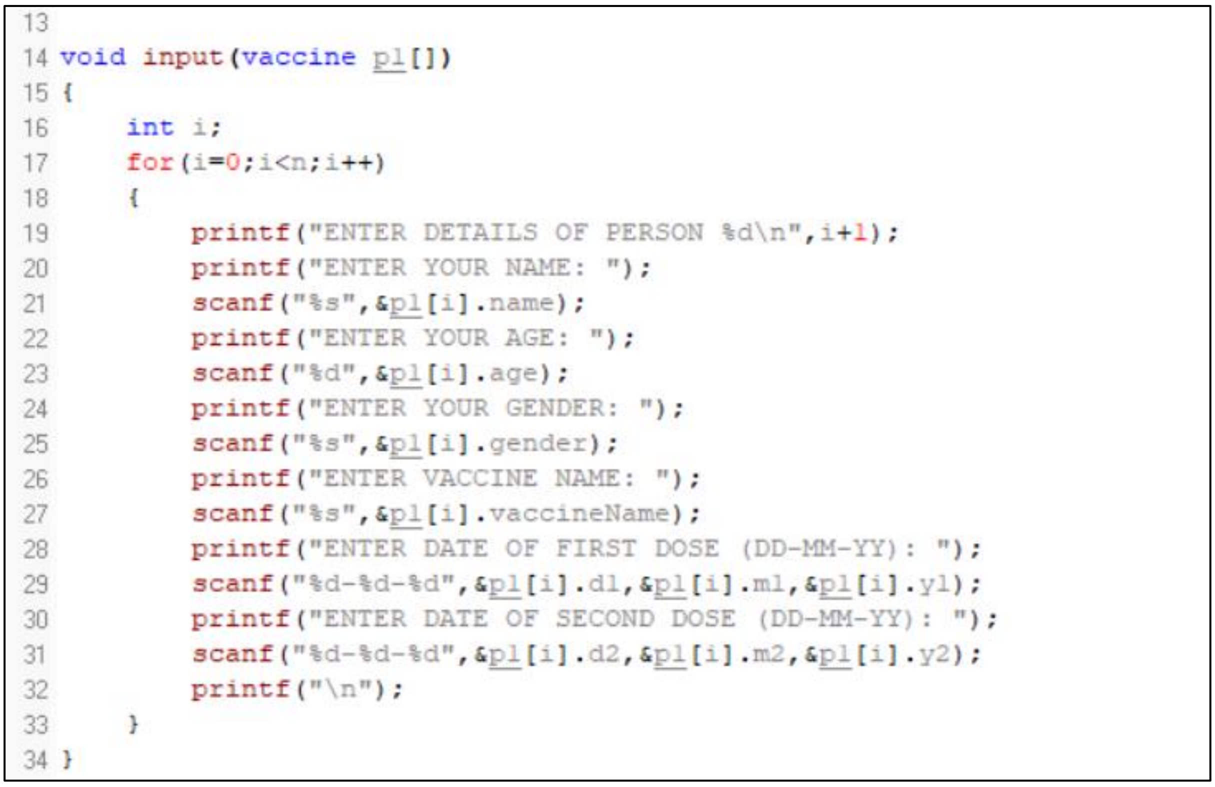
FLOWCHART


START
DEFINE A STRUCTURE VACCINE
char name[30]
int age
chargender[6]
char vaccineName[20]
int d1,m1,y1
int d2,m2,y2
INPUT N
DEFINE A POINTER
p1
DYNAMICALLY RESERVE MEMORY FOR N PERSONS
input(p1)
void input(vaccine p1[])
i = 0
i < N
No
RETURN()
Yes
INPUT p1[i].name
INPUT p1[i].age
INPUT p1[i].gender
INPUT p1[i].vaccineName
INPUT p1[i].d1,p1[i].m1,p1[i].y1
INPUT p1[i].d2,p1[i].m2,p1[i].y2
i++
output(p1)
END
void output(vaccine p1[])
i = 0
i < N
No
RETURN()
Yes
PRINT p1[i].name
PRINT p1[i].age
PRINT p1[i].gender
PRINT p1[i].vaccineName
PRINT p1[i].d1,p1[i].m1,p1[i].y1
PRINT p1[i].d2,p1[i].m2,p1[i].y2
i++

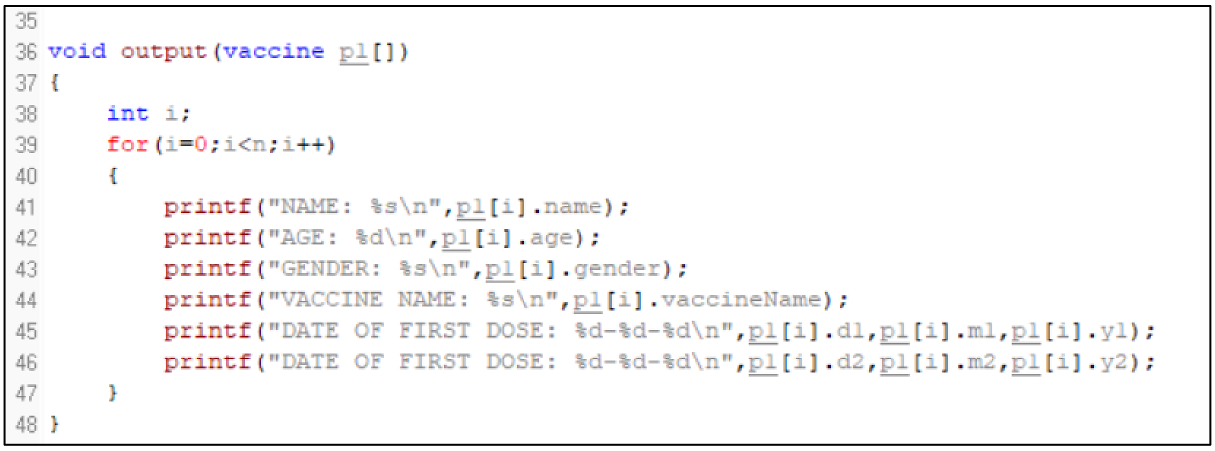
SOURCE CODE



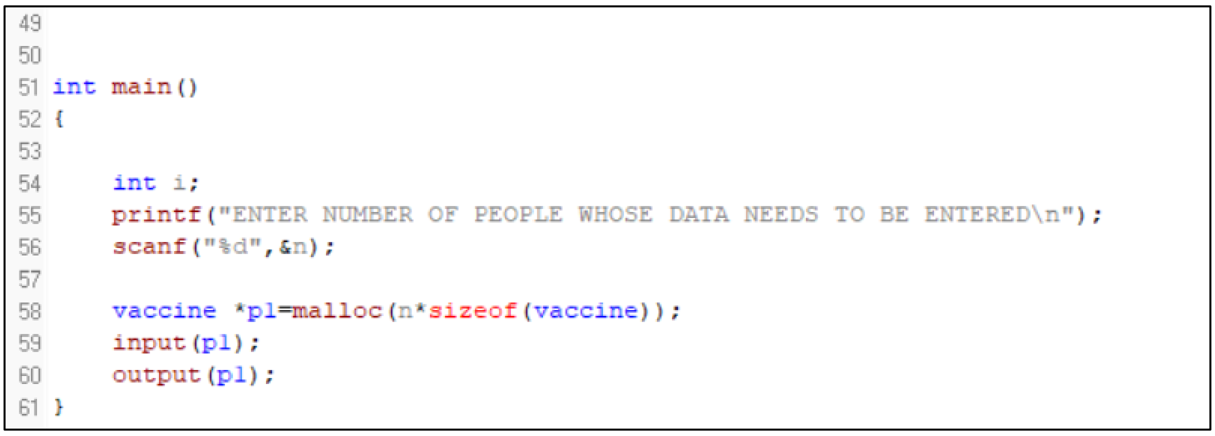
```c
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int n;
typedef struct vaccine{
    char name[30];
    int age;
    char gender[6];
    char vaccineName[20];
    int d1,m1,y1;
    int d2,m2,y2;
}vaccine;
```



```c

void input(vaccine p1[])
{
    int i;
    for(i=0;i<n;i++)
    {
        printf("ENTER DETAILS OF PERSON %d\n",i+1);
        printf("ENTER YOUR NAME: ");
        scanf("%s",&p1[i].name);
        printf("ENTER YOUR AGE: ");
        scanf("%d",&p1[i].age);
        printf("ENTER YOUR GENDER: ");
        scanf("%s",&p1[i].gender);
        printf("ENTER VACCINE NAME: ");
        scanf("%s",&p1[i].vaccineName);
        printf("ENTER DATE OF FIRST DOSE (DD-MM-YY): ");
        scanf("%d-%d-%d",&p1[i].d1,&p1[i].m1,&p1[i].y1);
        printf("ENTER DATE OF SECOND DOSE (DD-MM-YY): ");
        scanf("%d-%d-%d",&p1[i].d2,&p1[i].m2,&p1[i].y2);
        printf("\n");
    }
}
```

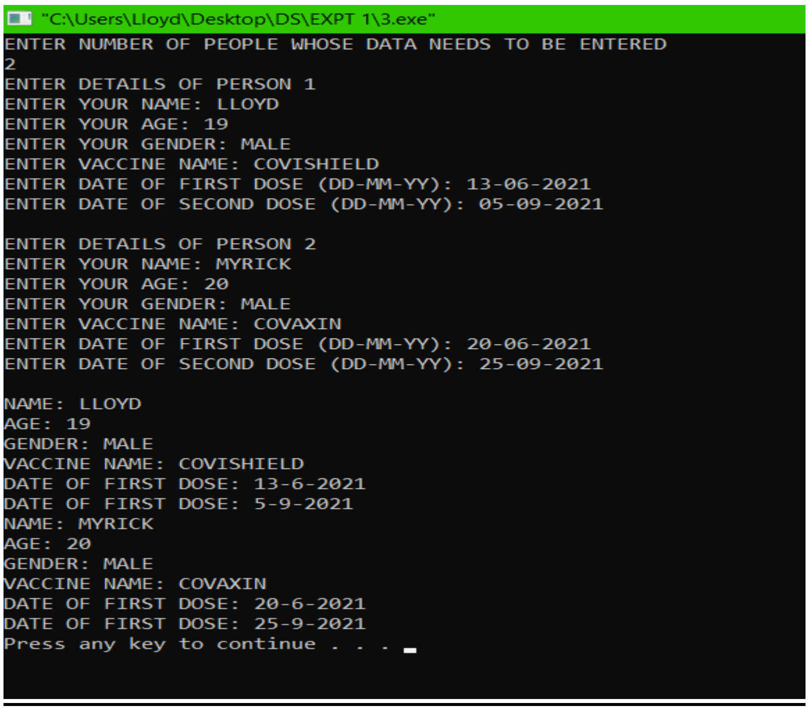


```c

void output(vaccine p1[])
{
    int i;
    for(i=0;i<n;i++)
    {
        printf("NAME: %s\n",p1[i].name);
        printf("AGE: %d\n",p1[i].age);
        printf("GENDER: %s\n",p1[i].gender);
        printf("VACCINE NAME: %s\n",p1[i].vaccineName);
        printf("DATE OF FIRST DOSE: %d-%d-%d\n",p1[i].d1,p1[i].m1,p1[i].y1);
        printf("DATE OF FIRST DOSE: %d-%d-%d\n",p1[i].d2,p1[i].m2,p1[i].y2);
    }
}
```



```c


int main()
{

    int i;
    printf("ENTER NUMBER OF PEOPLE WHOSE DATA NEEDS TO BE ENTERED\n");
    scanf("%d",&n);

    vaccine *p1=malloc(n*sizeof(vaccine));
    input(p1);
    output(p1);
}
```

OUTPUT:

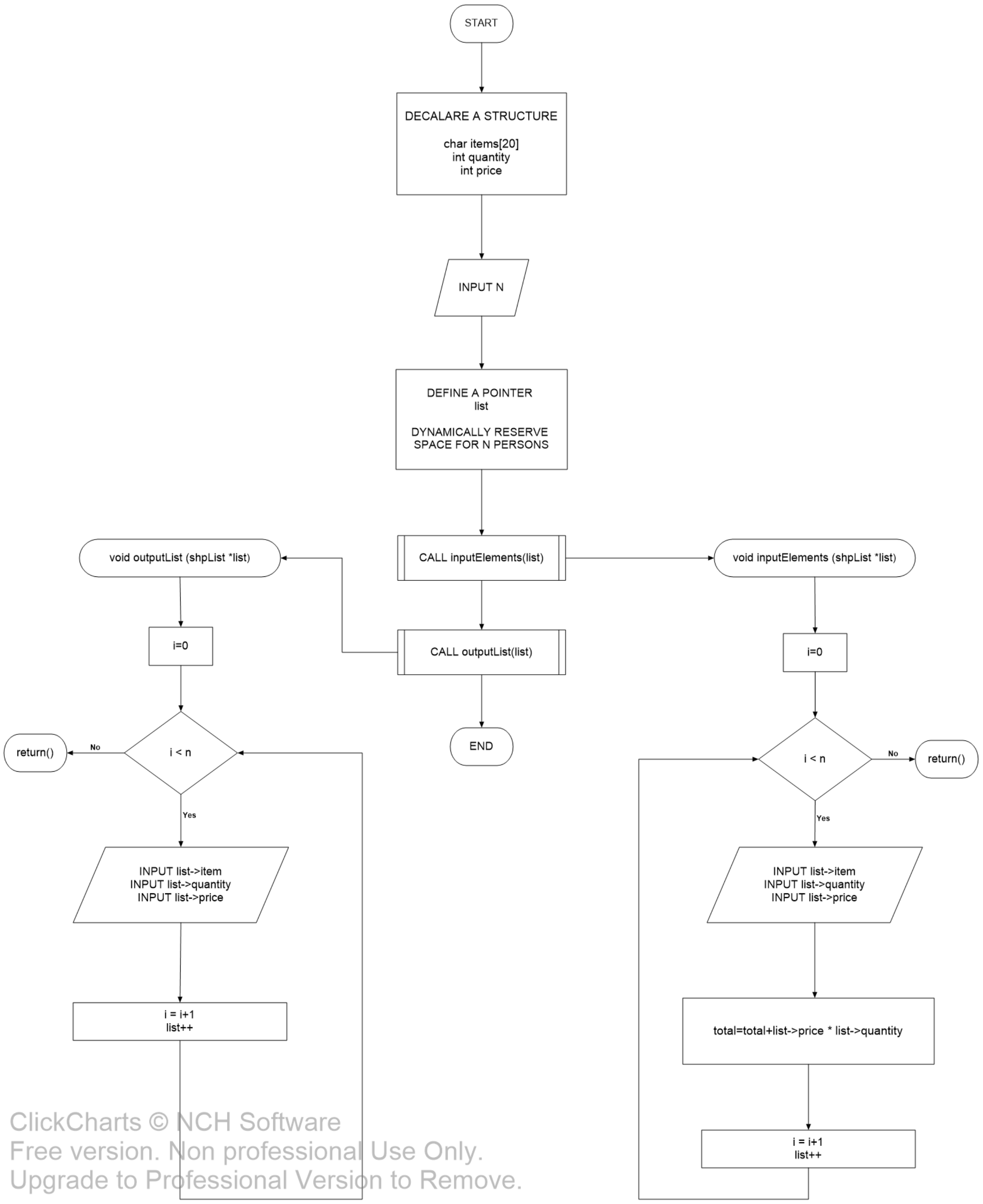


"C:\Users\Lloyd\Desktop\DS\EXPT 1\3.exe"

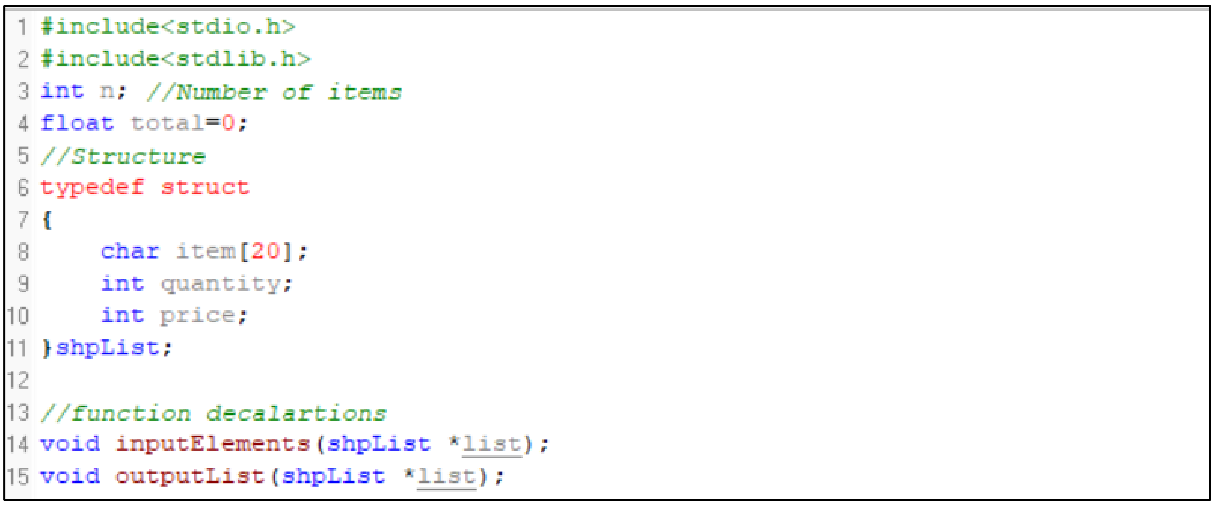
```
ENTER NUMBER OF PEOPLE WHOSE DATA NEEDS TO BE ENTERED
2
ENTER DETAILS OF PERSON 1
ENTER YOUR NAME: LLOYD
ENTER YOUR AGE: 19
ENTER YOUR GENDER: MALE
ENTER VACCINE NAME: COVISHIELD
ENTER DATE OF FIRST DOSE (DD-MM-YY): 13-06-2021
ENTER DATE OF SECOND DOSE (DD-MM-YY): 05-09-2021

ENTER DETAILS OF PERSON 2
ENTER YOUR NAME: MYRICK
ENTER YOUR AGE: 20
ENTER YOUR GENDER: MALE
ENTER VACCINE NAME: COVAXIN
ENTER DATE OF FIRST DOSE (DD-MM-YY): 20-06-2021
ENTER DATE OF SECOND DOSE (DD-MM-YY): 25-09-2021

NAME: LLOYD
AGE: 19
GENDER: MALE
VACCINE NAME: COVISHIELD
DATE OF FIRST DOSE: 13-6-2021
DATE OF FIRST DOSE: 5-9-2021
NAME: MYRICK
AGE: 20
GENDER: MALE
VACCINE NAME: COVAXIN
DATE OF FIRST DOSE: 20-6-2021
DATE OF FIRST DOSE: 25-9-2021
Press any key to continue . . .
```

5)

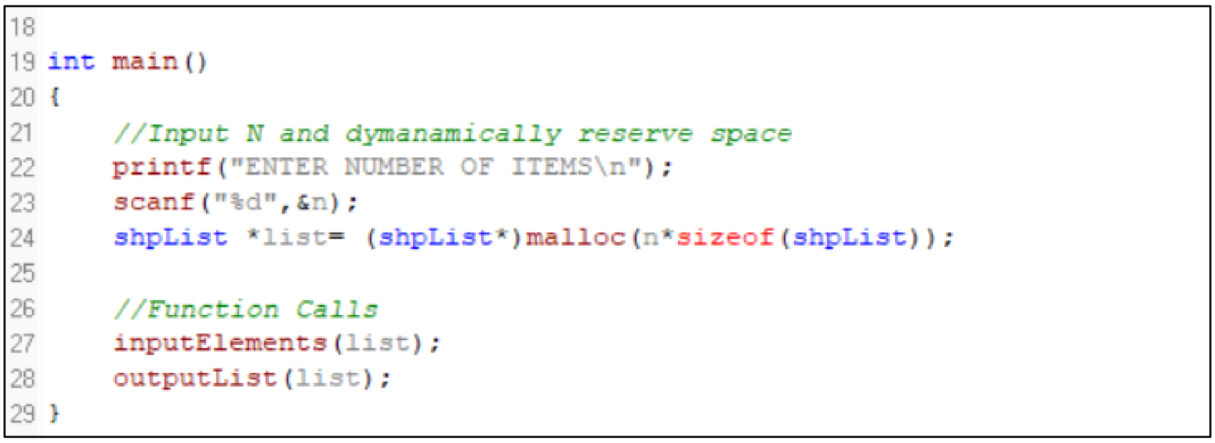
```
1. START
2.  DEFINE A STRUCTURE VACCIINE
        1. char item[
        2. int quantity
        3. char price
3. INPUT N
4. DECLARE A POINTER p1 OF TYPE VACCINE AND DYNAMICALLY  RESERVE MEMORY FOR N PERSONS
5. CALL inputElements(shpList *list)
6. CALL outputList(shpList *list)
7.END


Void inputElements(shpList *list)
1. for i=0,1,2,…n
        1.INPUT list->item, list->quantity, list->price
        2. total=total+( (list->quantity) * (list->price) )
        3. i = i + 1
        4. list++


Void outputList(shpList *list)
1. for i=0,1,2,…n
        1.OUTPUT list->item, list->quantity, list->price
        2. i = i + 1
        3. list++
```


START
DECALARE A STRUCTURE
char items[20]
int quantity
int price
INPUT N
DEFINE A POINTER
list
DYNAMICALLY RESERVE
SPACE FOR N PERSONS
CALL inputElements(list)
CALL outputList(list)
END
void outputList (shpList *list)
i=0
i < n
No
return()
Yes
INPUT list->item
INPUT list->quantity
INPUT list->price
i = i+1
list++
void inputElements (shpList *list)
i=0
i < n
No
return()
Yes
INPUT list->item
INPUT list->quantity
INPUT list->price
total=total+list->price * list->quantity
i = i+1
list++

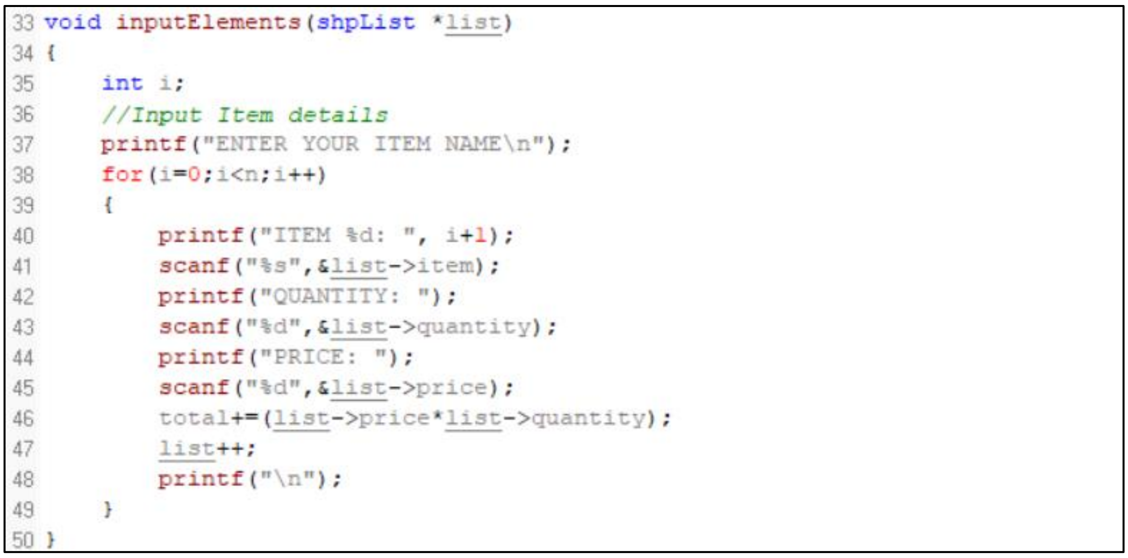
SOURCE CODE



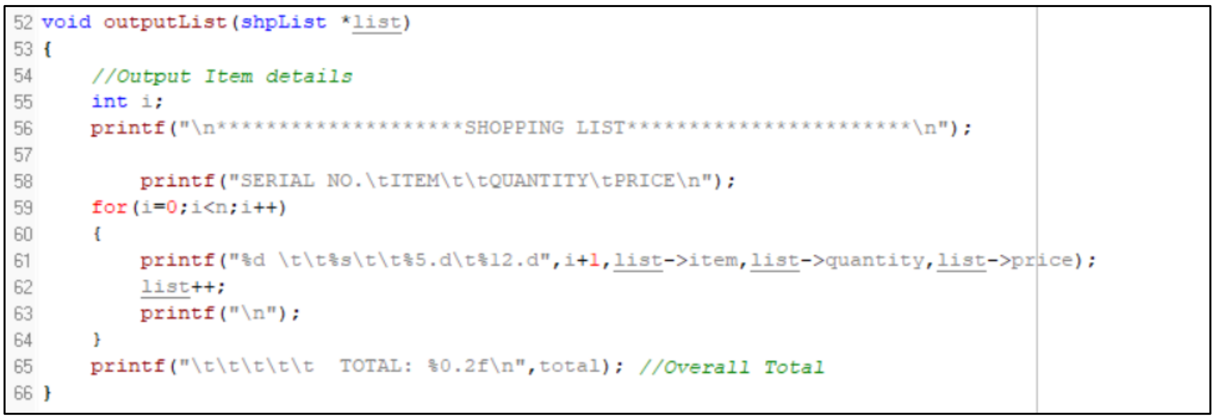
```c
#include<stdio.h>
#include<stdlib.h>
int n; //Number of items
float total=0;
//Structure
typedef struct
{
    char item[20];
    int quantity;
    int price;
}shpList;

//function decalartions
void inputElements(shpList *list);
void outputList(shpList *list);
```



```c

int main()
{
    //Input N and dymanamically reserve space
    printf("ENTER NUMBER OF ITEMS\n");
    scanf("%d",&n);
    shpList *list= (shpList*)malloc(n*sizeof(shpList));

    //Function Calls
    inputElements(list);
    outputList(list);
}
```

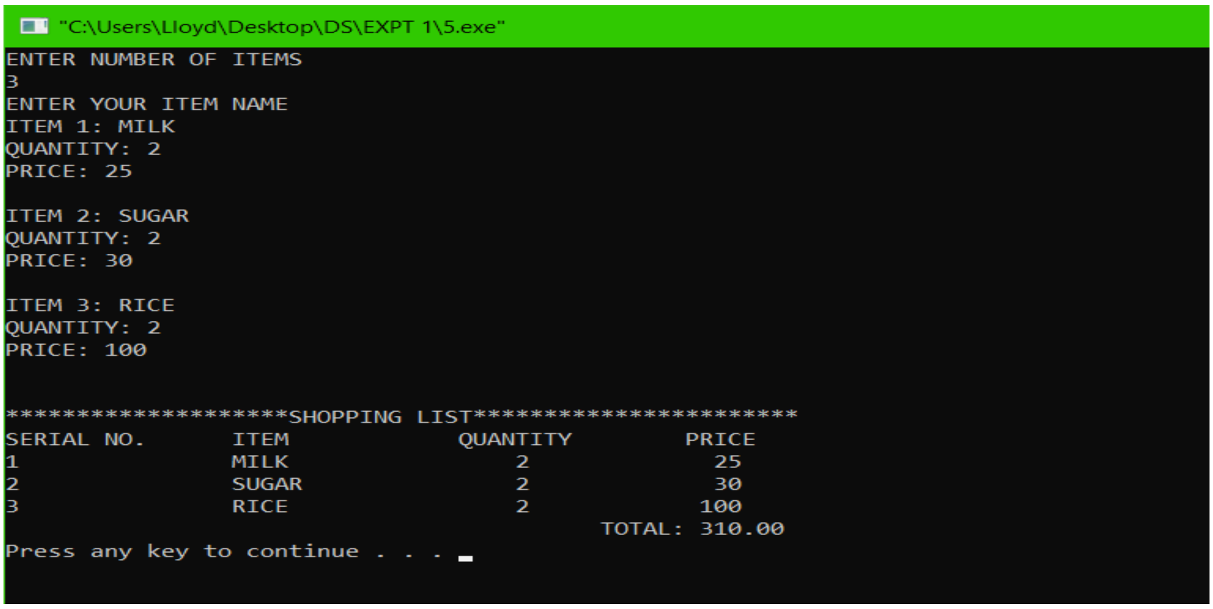


```c
void inputElements(shpList *list)
{
    int i;
    //Input Item details
    printf("ENTER YOUR ITEM NAME\n");
    for(i=0;i<n;i++)
    {
        printf("ITEM %d: ", i+1);
        scanf("%s",&list->item);
        printf("QUANTITY: ");
        scanf("%d",&list->quantity);
        printf("PRICE: ");
        scanf("%d",&list->price);
        total+=(list->price*list->quantity);
        list++;
        printf("\n");
    }
}
```



```c
void outputList(shpList *list)
{
    //Output Item details
    int i;
    printf("\n*******************SHOPPING LIST***********************\n");

        printf("SERIAL NO.\tITEM\t\tQUANTITY\tPRICE\n");
    for(i=0;i<n;i++)
    {
        printf("%d \t\t%s\t\t%5.d\t%12.d",i+1,list->item,list->quantity,list->price);
        list++;
        printf("\n");
    }
    printf("\t\t\t\t\t  TOTAL: %0.2f\n",total); //Overall Total
}
```

OUTPUT



"C:\Users\Lloyd\Desktop\DS\EXPT 1\5.exe"

```
ENTER NUMBER OF ITEMS
3
ENTER YOUR ITEM NAME
ITEM 1: MILK
QUANTITY: 2
PRICE: 25

ITEM 2: SUGAR
QUANTITY: 2
PRICE: 30

ITEM 3: RICE
QUANTITY: 2
PRICE: 100


********************SHOPPING LIST***********************
SERIAL NO.      ITEM            QUANTITY        PRICE
1               MILK                2            25
2               SUGAR               2            30
3               RICE                2           100
                                          TOTAL: 310.00
Press any key to continue . . . _
```

CONCLUSION AND FINDING

The given problem statements were successfully compiled and executed.