**PROBLEM:**

1.a) Find whether a triangle is equilateral/scalene/isosceles.

**AIM:**

To write a program whether a triangle is equilateral/scalene/isosceles.

**ALGORITHM:**

1. Start
2. Include all the required header files
3. Declare the variables s1,s2 and s3
4. Read the inputs s1,s2 and s3
5. Display the inputs s1,s2,s3
6. Ckeck IF( s1==s2 && s2==s3 && s3==s1),if the condition is true
7. Display”Equilateral triangle”
8. Check ELSE IF (s1==s2 || s1==s3 || s2==s3),if the condition is true
9. Display” Isosceles triangle”
10. Otherwise ELSE
11. Display “Scalene triangle”
12. Stop

**SOURCE CODE:**

#include<stdio.h>

int main()

{

int s1,s2,s3;

scanf("%d",&s1);

scanf("%d",&s2);

scanf("%d",&s3);

printf("Side 1:%d\n",s1);

printf("Side 2:%d\n",s2);

printf("Side 3:%d\n",s3);

if(s1==s2 && s1==s3 && s2==s3)

{

printf("Equilateral triangle");

}

else if(s1==s2 || s1==s3 || s2==s3)

{

printf("Isosceles triangle");

}

else

{

printf("Scalene triangle");

}

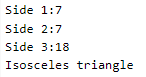
return 0;

}

**SAMPLE INPUT:**



**SAMPLE OUTPUT:**



**RESULT:**

Thus the program is verified successfully.

**PROBLEM:**

1.b) Get marks as input and print the grade

• Grade A-91-100

• Grade B-71-90

• Grade C-51-70

• Grade D-31-50

• Fail-less than 31.

**AIM:**

To create a C program to get a marks as input and print the grade.

**ALGORITHM:**

1. Start
2. Include all the required header files
3. Declare the variable mark
4. Read mark
5. Display mark
6. Check IF(mark>=91 && mark<=100),if the condition is true
7. Display”Grade A”
8. Check ELSE IF(mark>=71 && mark<=90),if the condition is true
9. Display”Grade B”
10. Check ELSE IF(mark>=51 && mark<=70),if the condition is true
11. Display”Grade C”
12. Check ELSE IF(mark>=31 && mark<=50),if the condition is true
13. Display”Grade D”
14. Check ELSE IF(mark<31),if the condition is true
15. Display”Fail”
16. Stop

**SOURCE CODE:**

#include<stdio.h>

int main()

{

int mark;

scanf("%d",&mark);

printf("Mark: %d\n",mark);

if(mark>=91 && mark<=100)

{

printf("Grade A");

}

else if(mark>=71 && mark<=90)

{

printf("Grade B");

}

else if(mark>=51 && mark<=70)

{

printf("Grade C");

}

else if(mark>=31 && mark<=50)

{

printf("Grade D");

}

else if(mark<31)

{

printf("Fail");

}

return 0;

}

**SAMPLE INPUT:**



**SAMPLE OUTPUT:**



**RESULT:**

Thus the program is verified successfully.

**PROBLEM:**

1 .c)Write a program to print the series 1,4,9,16,........Get the number of elements to be

printed as input.

**AIM:**

To create a program to print the series number.

**ALGORITHM:**

1.Start

2.Include all the required header files

3. Declare the variable num

4.Read the input num

5.Initialize i=1

6.Check WHILE (i<=num ),if the condition is true

7.Then display i\*i

8.Increment i

9.Stop

**SOURCE CODE:**

#include <stdio.h>

int main()

{

int num,i=1;

printf("Enter the number:");

scanf("%d",&num);

printf(

while (i<=num)

{

printf("%d ",i\*i);

i++;

}

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT**:



**RESULT:**

Thus the program is verified successfully.

**PROBLEM:**

1.d) Write a program to print the factors of a number.

**AIM:**

To create a program to print the factors of a number**.**

**ALGORITHM:**

1.Start

2.Include all the required header files

3.Declare the variables num,i

4.Read the input num

5.Check FOR( i=1;i<=num;++i)

6.Start loop from 1 to num by incrementing 1 in each iteration.

**7**. Then use IF statement inside for loop to print factors of a number if (num%i==0)

then i is a factor of num.

8.stop

**SOURCE CODE:**

#include <stdio.h>

int main()

{

int num, i;

printf("Enter a positive number: ");

scanf("%d", &num);

printf("%d\n",num);

printf("Factors of %d are:\n", num);

for (i=1;i<=num;++i)

{

if (num%i==0)

{

printf("%d\n",i);

}

}

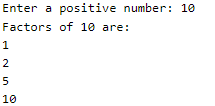
return 0;

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the program is verified successfully.

**PROGRAM**:

2.a) Program using function: Write a program to find factorial of the number using recursion.

**AIM:**

To create a program to find factorial of a number using recursion.

**ALGORITHM:**

1.Start

2.Read the variables a, fact

3.Declare the function, int factorial (a)

4.Call the function, factorial(a)

5.IF a= =1 or a= =0, return 1

6.ELSE fact =a\*factorial(a-1)

7.Display fact

8.Stop

**SOURCE CODE:**

#include<stdio.h>

int factorial(int);

int main ()

{

int a, fact;

printf("\nEnter a number: ");

scanf("%d",&a);

printf("%d\n",a);

fact =factorial(a);

printf("\nFactorial of %d is: %d",a, fact);

return 0;

}

int factorial(int a)

{

if(a==0||a==1)

{

return 1;

}

else

{

return(a\*factorial(a-1));

}

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the program is verified successfully.

**PROBLEM:**

2.b) Program using function: Write a program to swap numbers using function.

**AIM:**

To create a program to swap numbers using function.

**ALGORITHM:**

1.Start

2.Declare variable p,q,r

3.Read the variables p, q, r

4.Declare the function void swap (int p, int q, int r)

5.Call the function, swap (p, q, r)

6.Declare temp

7.Assign temp=p

8.Assign p=q

9.Assign q=r

10.Assign r=temp

11.Display p, q, r

12.Stop

**SOURCE CODE:**

#include <stdio.h>;

void swap(int p, int q,int r);

int main()

{

int p, q, r;

printf("Before Swapping:\n");

printf("p: ");

scanf("%d", &p);

printf("%d", p);

printf("\nq: ");

scanf("%d", &q);

printf("%d", q);

printf("\nr: ");

scanf("%d", &r);

printf("%d", r);

swap(p,q,r);

return 0;

}

void swap(int p, int q,int r)

{

int temp = p;

p = q;

q = r;

r = temp;

printf("\nAfter Swapping:");

printf("\np : %d", p);

printf("\nq : %d", q);

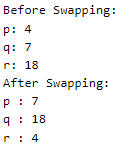
printf("\nr : %d", r);

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the program is verified successfully.

**PROBLEM:**

2.c) To calculate arithmetic operations using functions.

**AIM:**

To create a C program to calculate arithmetic operations using functions.

**ALGORITHM:**

1. Start
2. Include all the required header files
3. Declare the variables a and b
4. Read the inputs a and b
5. Display the inputs a and b
6. Call the add function
7. Declare the variables res
8. res=a+b
9. Display res
10. Call the sub function
11. res=a-b
12. Display res
13. Call the multi function
14. res=a\*b
15. Display res
16. Call the division function
17. res=a/b
18. Display res
19. Call the mod function
20. res=a%b
21. Display res

22.Stop

**SOURCE CODE:**

#include<stdio.h>

int add(int a,int b)

{

int res=a+b;

printf("Addition:%d\n",res);

}

int sub(int a,int b)

{

int res=a-b;

printf("Subtraction:%d\n",res);

}

int multi(int a,int b)

{

int res=a\*b;

printf("Multiplication:%d\n",res);

}

float division(float a,float b)

{

float res=a/b;

printf("Division:%f\n",res);

}

int mod(int a,int b)

{

int res=a%b;

printf("Modulus:%d\n",res);

}

int main()

{

int a,b;

scanf("%d",&a);

scanf("%d",&b);

printf("Num 1:%d\n",a);

printf("Num 2:%d\n",b);

add(a,b);

sub(a,b);

multi(a,b);

division(a,b);

mod(a,b);

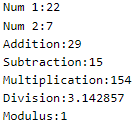
return 0;

}

**SAMPLE INPUT:**



**SAMPLE OUTPUT:**



**RESULT:**

Thus the program is verified successfully.

**PROBLEM:**

3.a) Given an array with 'n' elements, delete an element at a particular position in the array. Delete the element and displaying the updated array. Get the value of n, elements of array, element to be deleted as user input.

**AIM:**

To create a C program to delete an element at a particular position in the array.

**ALGORITHM:**

1. Start
2. Include all the required header files
3. Declare the variable n
4. Read the input n
5. Read the array elements of arrays arr[n]
6. Display the array elements of arrays arr[n]
7. Declare the variables del
8. Read the input del
9. Initialize the variable k=0,count=0
10. Initialize the variable i=0
11. Initialize temp[n]
12. Check FOR condition, if the condition is true iterate the loop for n times
13. Check IF Condition,if(arr[i]!=del)
14. temp[k]=arr[i]
15. Increment the k value
16. Otherwise ELSE condition
17. Increment the count value
18. Finally check IF condition,if(count!=0)
19. Check FOR condition,if the condition is true iterate the loop for k times
20. Display temp[i]
21. Otherwise ELSE condition
22. Display "Element dosen't found in the array"
23. Stop

**SOURCE CODE:**

#include<stdio.h>

int main()

{

int n,i,del,k=0,count=0;

scanf("%d",&n);

int arr[n],temp[n];

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

{

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

}

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

{

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

}

scanf("%d",&del);

printf("\nDelete element:%d",del);

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

{

if(arr[i]!=del)

{

temp[k]=arr[i];

k++;

}

else

{

count+=1;

}

}

if(count!=0)

{

printf("\nArray after deletion:\n");

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

{

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

}

}

else

{

printf("Element dosen't found in the array");

}

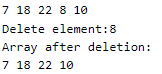
return 0;

}

**SAMPLE INPUT:**



**SAMPLE OUTPUT:**



**RESULT:**

Thus the program is verified successfully.

**PROBLEM:**

3b). Write a program to find whether the given string is a palindrome or not without using string library functions.

**AIM:**

To write a program to check the given string is a palindrome or not without using string library functions.

**ALGORITHM:**

1. Start
2. Include all the head files
3. Declare two character array a[50],str[50]
4. Declare three integer i,k=0,c
5. Read a
6. Initialize a FOR loop with int i=0 ,condition a[i]!=’\0’ with increment i++
7. Initialize a FOR loop with int j=i-1 ,condition j>=0 with increment j++
8. Iterate ‘a’ and store it in ‘str’
9. Initialize a FOR loop with int i=0 ,condition a[i]!=0 with increment i++
10. Iterating to check whether the each character in ‘a’,’str’ are same
11. If two are same upgrade the c=1 else c=0
12. IF c==1 print palindrome
13. ELSE if c==0 print not a palindrome
14. Stop

**SOURCE CODE:**

#include<stdio.h>

#include<string.h>

int main()

{

char a[50],str[50];

int i,k=0,c;

scanf("%s",a);

for( i=0;a[i]!='\0';i++);

for(int j=i-1;j>=0;j--)

{

str[k]=a[j];

k++;

}

for( i=0;a[i]!='\0';i++)

{

if(a[i]==str[i])

{

c=1;

}

else

{

c=0;

break;

}

}

if(c==1)

{

printf("Palindrome");

}

else

{

printf("Not a Palindrome");

}

return 0;

}

**SAMPLE INPUT:**



**SAMPLE OUTPUT:**



**RESULT:**

Thus the program is verified successfully.

**PROBLEM:**

3.c) To create a program using strcat(),strcmp(),strcpy(),strlen(),strrev(). Get two strings as input , perform the above mentioned methods and print the result.

**AIM:**

To create a program using strcat(),strcmp(),strcpy(),strlen(),strrev() using two strings input.

**ALGORITHM:**

1. Start
2. Include all the head files
3. Include string.h header file to perform string operations
4. Declare two char array a[50],b[50],s,c[50]
5. Read a and b
6. Initialize l which is length of a and l1which is the length of b using strlen() function
7. Print the copy of a which is stored in c using strcopy()
8. Print the result of the concatenation using strcat()
9. Print l and l1
10. Compare a and b using strcmp()
11. If s==0 print strings are same
12. Else strings are not same
13. Stop

**SOURCE CODE:**

#include<stdio.h>

#include<string.h>

int main()

{

char a[50],b[50],s,c[50];

scanf("%s",a);

scanf("%s",b);

int l,l1;

l=strlen(a);

l1=strlen(b);

printf("%s\n",strcpy(c,a));

printf("%s\n",strcat(a,b));

printf("%d\n",l);

printf("%d\n",l1);

s=strcmp(a,b);

if(s==0)

{

printf("Strings are same\n");

}

else

{

printf("Strings are not same");

}

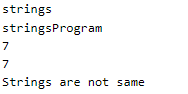
return 0;

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**



**RESULT:**

Thus the program is verified successfully.

**PROBLEM:**

3.d) Write a program to create a C program for matrix addition, subtraction using array.

**AIM:**

To write a program for matrix addition, subtraction using array.

**ALGORITHM:**

1. Start
2. Include all the required header files
3. Read n
4. Read the array elements of arrays a[n], b[n]
5. Declare the variables sum, diff
6. Initialize the variable i=0
7. Check FOR condition, if the condition is true iterate the loop for n times
8. sum=a[i] + b[i]
9. Display sum
10. Initialize the variable i=0
11. Check FOR condition, if the condition is true iterate the loop for n times
12. diff=a[i]-b[i]
13. Display diff
14. Stop

**SOURCE CODE:**

#include<stdio.h>

int main(){

int n;

scanf("%d",&n);

int a[n],b[n];

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

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

}

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

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

}

int sum;

int diff;

printf("Addition:");

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

sum=a[i]+b[i];

printf("%d ",sum);

}

printf("\n");

printf("Subtraction:");

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

diff=a[i]-b[i];

printf("%d ",diff);

}

return 0;

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the program is verified successfully.

**PROBLEM:**

4.a) Write a program to create a C program for employee information system using nested structure.

**AIM:**

To create a C program for employee information system using nested structure.

**ALGORITHM:**

1. Start
2. Create a structure employee with the members name, id, sal and the nested structure dob which contains the members date, mon, yr.
3. Create the structure variables d for the structure dob and the variable e for the employee
4. Read the members of the structure
5. Display the members of the structure
6. Stop

**SOURCE CODE:**

#include<stdio.h>

struct employee{

char name[50];

int id;

int sal;

struct dob{

int date;

char mon[20];

int yr;

}d;

}e;

int main(){

scanf("%s",e.name);

scanf("%d",&e.id);

scanf("%d",&e.sal);

scanf("%d",&e.d.date);

scanf("%s",e.d.mon);

scanf("%d",&e.d.yr);

printf("Name:%s\n",e.name);

printf("ID:%d\n",e.id);

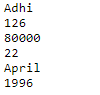
printf("Salary:%d\n",e.sal);

printf("DOB:%d-%s-%d",e.d.date,e.d.mon,e.d.yr);

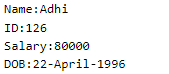
return 0;

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the program is verified successfully.

**PROBLEM:**

4.b) Write a C program to find the modulus of two numbers using pointers.

**AIM:**

To write a program to find the modulus of two numbers using pointers.

**ALGORITHM:**

1. Start
2. Include all the header files which are used in the program
3. Take the quotient and reminder from the user as a and b
4. Calulate modulus of a and b and store it in the variable m
5. Store the address of m in mod
6. Display \*mod
7. Stop

**SOURCE CODE:**

#include <stdio.h>

int main(){

int a,b,m,\*mod;

scanf("%d",&a);

scanf("%d",&b);

m=a%b;

mod=&m;

printf("Modulus : %d",\*mod);

return 0;

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the program is verified successfully.

**PROBLEM:**

4.c) Create a program containing the union student with the following members and display

them.

* Roll Number
* Name
* Average
* Grade

**AIM:**

To write a program containing the union student with the given members and display

them.

**ALGORITHM:**

1. Start
2. Include all the header files which are used in the program
3. Declare the union student and include the members rn, name, mark, avg and grad
4. Collect the data such as roll number and name from the user and display it using union variable immediately
5. Collect the marks of five subject from the user and calculate the average and display it using the union variable
6. Then calculate the grade using average. If the average is greater than 70, grade is 1. Else if greater 49 and less than 71 grade is 2. Else grade is 3.
7. Display the grade using the union variable.
8. Stop

**SOURCE CODE:**

#include<stdio.h>

union student{

int rn;

char name[50];

int mark[5];

float avg;

int grad;

};

int main()

{

int total;

union student s;

scanf("%d",&s.rn);

printf("Roll.no: ");

printf("%d\n",s.rn);

scanf("%s",s.name);

printf("Name: ");

printf("%s\n",s.name);

for(int i=0;i<5;i++){

scanf("%d",&s.mark[i]);

total=total+s.mark[i];

}

s.avg=total/5;

printf("Average: ");

printf("%f\n",s.avg);

printf("Grade: ");

if(s.avg>70){

s.grad=1;

printf("%d",s.grad);

}

else if(s.avg>=50 && s.avg<=70){

s.grad=2;

printf("%d",s.grad);

}

else{

s.grad=3;

printf("%d",s.grad);

}

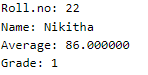
return 0;

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the program is verified successfully.

**PROBLEM:**

4.d) Create a program to swap the number using function with call by reference.

**AIM:**

To write a program to swap two numbers.

**ALGORITHM:**

 1. Start.

2. Get a and b.

3. Declare the function void swap (int\* a,int \*b)

4. call the function swap(&a,&b).

5. Assign temp=\*a.

6. Assign \*a=\*b.

7. Assign \*b=temp.

8. Print a and b.

9. Stop.

**SOURCE CODE:**

#include <stdio.h>

void swap(int \*a,int \*b);

int main()

{

int a,b;

scanf("%d%d",&a,&b);

swap(&a,&b);

printf("Inside main:%d %d",a,b);

return 0;

}

void swap(int \*a,int \*b)

{

int temp;

temp=\*a;

\*a=\*b;

\*b=temp;

printf("Inside swap:%d %d\n",\*a,\*b);

}

**SAMPLE INPUT:**



**SAMPLE OUTPUT:**



**RESULT:**

Thus the program is verified successfully.

**PROBLEM:**

1. Write a program to implement a stack using Linked List.

**AIM:**

To create a program to implement a stack using Linked List.

**ALGORITHM:**

1. start
2. Include all the **header files** which are used in the program
3. Declare all the **user defined functions**.
4. Create a node pointer head and make it as NULL
5. Read X
6. Call the insert function to insert the elements at front.
7. Create the node and insert node data and its address as NULL

8.Check if the stack is empty or not.

9.If it is e**mpty**, then set **newnode -> next = NULL**

10.If it is n**ot empty**, then set **newnode -> address** = head

**11.**Finally, set head = **newnode.**

12.To display the stack call the function display

13.Check if the stack is empty or not

14.If it is e**mpty,** then display **stack is empty** and terminate the function.

15.If it is not empty, display head->address until head!=NULL

16.Stop

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node\*addr;

};

void insert(struct node\*\*q,int val)

{

struct node\*nn=(struct node\*)malloc(sizeof(struct node));

nn->data=val;

nn->addr=NULL;

if(\*q==NULL)

{

\*q=nn;

}

else

{

nn->addr=(\*q);

(\*q)=nn;

}

}

void display(struct node\*q)

{

if(q==NULL)

{

printf("Stack is empty");

}

else

{

while(q!=NULL)

{

printf("%d\t",q->data);

q=q->addr;

}

}

}

int main()

{

struct node\*head=NULL;

int x;

while(1)

{

scanf("%d",&x);

if(x>0)

{

insert(&head,x);

}

else

{

break;

}

}

display(head);

}

**SAMPLE INPUT :**

****

**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the program is verified successfully.

**PROBLEM:**

10.Write a program to implement a queue using Arrays.

**AIM:**

To create a program to implement queue using arrays.

**ALGORITHM:**

1. Start.

2. Include all the head files.

3. Declare the function void display(int r,int\* arr).

4. Declare the variables n,r=-1,f=-1,element.

5. Read n.

6. Initialize a for loop with int i=0 ,condition i<n with increment i++.

7. If r == n-1 print Queue is full.

8. Else if f == -1 declare f=0.

9. Increment r by 1.

10. Assign arr[r]=element.

11. Print Queue elements are.

12. Call the function void display(r,arr).

13. If r == -1 print Queue is empty.

14. Else initialize a for loop with int i=0,condition i< = r with increment i++.

15. Print arr[i].

16.Stop.

**SOURCE CODE:**

#include <stdio.h>

void display(int r,int\* arr);

int main(){

int n,r=-1,f=-1,element;

scanf("%d",&n);

int arr[n];

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

scanf("%d",&element);

if(r==(n-1)){

printf("Queue is full \n");

}

else{

if(f==-1){

f=0;

}

r++;

arr[r]=element;

}

}

printf("Queue elements are :\n");

display(r,arr);

return 0;

}

void display(int r,int\* arr){

if(r==-1){

printf("\n Queue is empty");

}

else{

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

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

printf(" ");

}

}

}

**SAMPLE INPUT:**



**SAMPLE OUTPUT:**



**RESULT:**

Thus the program is verified successfully.

**PROBLEM:**

11. Write a program to implement a queue using Linked List.

**AIM:**

To implement a queue using using linked list.

**ALGORITHM:**

1. Start.
2. Include all the required header files
3. Declare all user defined functions
4. Define node with two members data and addr
5. Read num
6. Call insert function and insert elements at the end
7. Create the node and insert the node data and its addr as NULL
8. If queue is empty then head=newnode
9. Else store head in temp
10. While temp->addr!=NULL
11. Temp=temp->addr
12. Temp->addr=newnode
13. To display the function , call the display function
14. Display head->addr until head!=NULL
15. Stop

**SOURCE CODE:**

#include <stdio.h>

#include <cstdlib>

struct node

{

int data;

struct node\* addr;

};

void insertatend(struct node\*\* q,int value)

{

struct node\* newnode;

newnode=(struct node\*)malloc(sizeof(struct node));

newnode->data=value;

newnode->addr=NULL;

if((\*q)==NULL)

{

(\*q)=newnode;

}

else

{

struct node\* temp=(\*q);

while(temp->addr!=NULL)

{

temp=temp->addr;

}

temp->addr=newnode;

}

}

void display(struct node\* q)

{

while(q!=NULL)

{

printf("%d",q->data);

printf(" ");

(q)=(q)->addr;

}

}

int main()

{

//let num be the numbers in the list

struct node\* head=NULL;

int num;

while(true)

{

scanf("%d",&num);

if(num>=0)

{

insertatend(&head,num);

}

else

{

break;

}

}

printf("Implementation of Queue using linked list :\n");

display(head);

return 0;

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**



**RESULT:**

Thus the program is verified successfully.