Ex. No : 1	TOWER OF HANOI
Date : 17/10/22	

Aim :

To create a program to Tower of Hanoi using accursion.

Algorithm :

Step 1 : Start the process

Step 2 : Declare the variable

Step 3: Create void Hanoi function add find variables

Step 4: Include character : 'A' = 'B' = 'D' = 'C' to the disk

Step 5 : Find the if statement if (b1 = 0) find the how many disk in the Hanoi.

Step 6 : Stop the process.

TOWER OF HANOI

```
#include<iostream.h>
#include<conio.h>
void main()
{
int n;
clrscr();
cout<<"\t\t\t\t TOWER OF HANOI\n";
cout<<"\t\t\t\t ***** ** *****\n" ;
cout<<"Enter the number of disk:";
cin>>n;
void tower(int,char,char,char);
tower(n,'A','B','C');
getch();
}
void tower(int a,char from,char aux,char to)
{
if(a==1)
{
cout<<"\t  Move disk 1 From"<<from<<" to"<<to<<"\n\n";
}
else
{
tower(a-1,from,to,aux);
cout<<"\t Move Disk"<<a<<" from"<<from<<" to"<<to<<"\n\n";
```

```
tower(a-1,aux,from,to);
}
}
```

OUTPUT

TOWER OF HANOI

***** ** *****

Enter the number of disk:4

Move disk 1 FromAtoB

Move Disk2fromAtoC

Move disk 1 FromBtoC

Move Disk3fromAtoB

Move disk 1 FromCtoA

Move Disk2fromCtoB

Move disk 1 FromAtoB

Move Disk4fromAtoC

Move disk 1 FromBtoC

Move Disk2fromBtoA

Move disk 1 FromCtoA

Result:

Thus the program is successfully executed and output is verified.

Ex. No : 2	BINARY SEARCH
Date : 19/10/22	

Aim :

To create a program to traversal through binary Search tree using traversal.

Algorithm :

Step 1 : Start the process

Step 2: Declare the class binary search tree with data member and number function

Step 3: Create empty() function using return value of binary search tree.

Step 4: To Insert or functions, in order() postorder and preorder() function declared

Step 5: Using pointer variable to the tree from and create class from derived class

Step 6: the main function () to Insert, values for tree order

Step 7: Stop the process.

BINARY SEARCH

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
int a[50],n,i,j,max,bin,mid,min=1,n1,temp;
clrscr();
cout<<"\n\t BINARY SEARCH";
cout<<"\n\t Enter the number of terms:";
cin>>n;

max=n;
cout<<"Enter the number one by one:";
for(i=1;i<n;i++)
{
cin>>a[i];
}
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
if(a[i]>a[j])
{
temp=a[i];
a[i]=a[j];
a[j]=temp;
}
}
}
cout<<"ASCENDING NUMBER:";
for(i=0;i<n;i++)
{
cout<<a[i]<<"\n";
}
cout<<"\n\t Enter the number to search:";

cin>>n1;
cout<<"\n\t linear search";
for(i=0;i<=n;i++)
{
if(n1==a[i])
{
cout<<"\n\t The number is:"<<a[i];
cout<<"\n\t The position of the number:"<<i;
g
```

```
}
}
cout<<"\n Number not found";
bin:cout<<"\n Binary search";
while(max>=min)
{
mid=(min+max)/2;
if(n1==a[mid])
{
cout<<"Number is:"<<a[mid];

cout<<"Position of the number is:" <<mid;
getch();
exit(0);

}
else if(n1<a[mid])
max=mid-1;
else
min=mid+1;
}
cout<<"Number not found";
getch();
}
```

OUTPUT:

```
Binary search
 Enter the number of terms:7

Enter the numbers one by one:
22
33
65
98
55
14

78

 Ascending numbers:
1
14
22
33
55
65
78

Enter the number to search:55

Linear search
The number is:55
The position of the number:4
Binary search
Number is:55
Position of the number is:4
```

Result:
Thus the program is successfully executed and output is verified.

Ex. No : 3	STACK USING LINKED LIST.
Date : 21/10/22	

Aim :

To write a c++ program to perform various operations on stack using linked list

Algorithm :

Step 1: Start the program

Step 2: Declare the object and the class

Step 3: Declare the base class structure

Step 4: Declare the function. push () pop () and show ()

Step 5: Create the base class object and pointer variable.

Step 6: Declare the switch case statement and insert some operations.

Step 7: Executed and display the result.

Step 8: Stop the program.

3. STACK USING LINKED LIST

```
#include<iostream.h>
#include<conio.h>
void main()
{
int a[5],j,k,i,item,top=0,b,n;

clrscr();
cout<<"\n\t stack operation";
cout<<"\n\t ***************";
cout<<"\n\t enter the size of stack:";
cin>>n;

b:

cout<<"\n\n 1.push\n\n 2.pop\n\n 3.list of stack\n\n 4.exit:"; cout<<"\n\n what
you want to prefer:"; cin>>j;

switch(j)
{
case 1:
cout<<"\n enter the elements do you want to insert:";
cin>>item;

if(top==n)
{

cout<<"\n stack is full";
goto b;
}
do
{

top=top+1;
a[top]=item;
}

while(j==2||j==3);
goto b;
```

```
case 2:
if(top<=0)
{

cout<<"stack is empty";
goto b;
}
do

{
k=a[top];top=top-1;
cout<<"\n deleted item is \n"<<k;
}
while(j==1||j==3);
goto b;
case 3:

cout<<"\n list of stack\n\n";

for(i=1;i<=top;i++)
cout<<a[1];
if(j!=4)
goto b;
}
getch();
}
```

OUTPUT:

```
stack operation
***************
enter the size of stack:8
1.push
2.pop
3.list of stack
4.exit:
what you want to prefer:1
enter the elements do you want to insert:21
1.push
2.pop
3.list of stack
4.exit:
what you want to prefer:2
deleted item is 21
1.push
2.pop
3.list of stack
4.exit:
what you want to prefer:3
list of stack
1.push
2.pop
```

3.list of stack

4.exit:

what you want to prefer:4

Result:

Thus the program is successfully executed and output is verified.

Activate Windows
Go to Settings to activate Windows.

Ex. No : 4	CIRCULAR QUEUE
Date : 26/10/22	

Aim :

To write a C++ program using to perform various operation in Circular Queue.

Algorithm :

Step 1 : Start the process

Step 2: Declare the base class Circular Queue.

Step 3: Declare and define the function

Step 4: Declare and defined the display() enqueue() and delete element()

Step 5 : Create the base class from derived class.

Step 6 : Declare the switch case statement and value loop for condition checking

Step 7: Display the result.

Step 8 : Stop the program.

CIRCULAR QUEUE

```
#include<iostream.h>
#include<stdio.h>
#include<conio.h>
struct node
{
int data;
node *next;
}
*front=NULL,*rear=NULL,*p=NULL,*np=NULL;
void push(int x)
{

np=new node;
np->data=x;
np->next=NULL;
if(front==NULL)
{
front=rear=np;
rear->next=NULL;
}
else
{
rear->next=np;
rear=np;
rear->next=NULL;
}
}
int remove()
{
int x;
if(front==NULL)
{
cout<<"empty queue \n";
}
else
{

p=front;
x=p->data;
front=front->next;
delete(p);
return(x);
}
}
```

```
int main()
{

int n,c=0,x,true;
clrscr();
cout<<"enter the no of values to be pushed into queue \n";
cin>>n;
while(c<n)
{
cout<<"enter the value to be entered into queue \n";
cin>>x;
push(x);
c++;
cout<<"\n\n removed value";
while(true)
{

if(front!=NULL)
cout<<remove()<<endl;
else
break;
}
getch();
}
}
```

OUTPUT:

```
Enter the no of values to be pushed into queue
5

Enter the value to be entered into queue
12
Removed value12
```

Result:

Thus the program is successfully executed and output is verified.

Ex. No : 5	QUICK SORT.
Date : 27/10/22	

Aim :

To write a c++ program to sort an array to an element using quick sort.

Algorithm :

Step 1 : Start the program

Step 2 : Declare the variables

Step 3 : Define and find the while loop condition

Step 4 : Create some objects

Step 5 : Enter the element using space operation

Step 6 : Execute the program.

Step 7 : Display the result.

Step 8 : Stop the program.

QUICK SORT

```
#include<iostream.h>
 #include<conio.h>
 int part(int low,int high,int*a)
 {
 int i,h=high,l=low,p,t;
 p=a[low];
 while(low<high)
 {
 while(a[l]<p)
 {
 l++;

 }
 while(a[h]>p)
 {
 h--;
 }
 if(l<h)
 {
 t=a[l];
 a[l]=a[h];
 a[h]=t;
 }
 else
 {
 t=p;
 p=a[l];
 a[l]=t;
 break;
 }
 }
 return h;
 }
 void quick(int l,int h,int*a)
 {
 int index,i;

 if(l<h)
 {
 index=part(l,h,a);
 quick(l,index-1,a);
 quick(index+l,h,a);
 }
```

Activate Windows
Go to Settings to activate Windows.


```
}
int main()
clrscr();
int a[100],n,l,h,i;
cout<<"Enter no of element:";
cin>>n;

cout<<"Enter the element(use as a seperator)";
for(i=0;i<n;i++)
cin>>a[i];
cout<<"\n Initial array\n";
for(i=0;i<n;i++)
{
cout<<a[i]<<"\t";
}
n=n-1;

l=0;
quick(l,h,a);
cout<<"\n After sorting:\n";
for(i=0;i<n;i++)
{
cout<<a[i]<<"\t";
}
getch();
return 0;
}
```

OUTPUT:

```
Enter no of element:7
Enter the element(use as a seperator)
12
13
14
15
16
17
18

Initial array :
12    13    14    15    16    17    18

After sorting:
12    13    14    15    16    17    18
```

Result:

Thus the program is successfully executed and output is verified.

Ex. No : 6	HEAP SORT.
Date : 28/10/22	

Aim :

To create a program to solve of elements in ascending order using heap sort.

Algorithm :

Step 1: Start the program.

Step 2: Declare the class as prime with data number and member function.

Step 3: Consider the argument constructor prime or with integer argument.

Step 4: To call the function Calculate () and to the following steps.

Step 5: Check whether the IC value in Increments ble is as 1.

Step 6: If it is then display the value

Step 7: Stop the program

HEAP SORT

```
#include <iostream.h>
#include<conio.h>
Voidmax-heapify(int*a,inti,int n)
{
Intj,temp;
Temp=a[i];
J=2*I;
While(j<=n)
{
If(j<n.&&a[j+1]>a[j])
J=j+I;
If(temp>a[j])
Break;
Else if(temp<=a[j])
{
A[j/2]=a[j];
J=2*j;
}
}
A[j/2=temp;
Return;

Voidheapsort(int*a,int n)
{
Int I, temp;
For(i=n;i>2;i--)
{
Temp=a[i];
A[i]=a[1];
A[1]=temp;
Max-heapify(a,1,i-1);
}
}
Voidbuild-maxheap(int#a,int n)
{
Int I;
For(i=n/2;i>=1;i--);
{
Max-heapify(a,I,n);
}
}
Int main()
{
Int n,I,x;
Cout<<"enter no of elements of array\n:;
Cin>>n;
Int a[20];
For(i=1;i<-n;i++)
{
```

```
Cout<<"enter element"<<(i)<<end1;
Cin>>a[i];

Build-maxheap(a,n);
Heapsort(a,n);
Cout<<"sorted output"|n;
{
Cout<<a[i<<end1;

Getch();
Return(0);
```

OUTPUT:

```
Enter no of elements of array
7
Enter element1
34
Enter element2
45
Enter element3
12
Enter element4
40
Enter element5
6
Enter element6
75
Enter element7
36

Sorted output
0
6
12
36
40
45
75
```

Result:

Thus the program is successfully executed and output is verified

Ex. No : 07	KNAPSACK PROBLEM
Date : 31/10/22	

Aim :
To create a program to solve the knapsack problem using greedy method.

Algorithm :

Step 1: Start the program
Step 2: Declare the variable
Step 3: Declare and define the class knapsack
Step 4: Create and define the base class object
Step 5: Declare and define the get() function find f(), findout (), display() function
Step 6: Execute the program
Step 7: Display the result
Step 8: Stop the program

KNAPSACK PROBLEM

```
#include<iostream.h>
#include<conio.h>
void main()
{
float p[100],w[100],x[10],profit=0,u;
int n,i;
clrscr();
cout<<"\t\t KNAPSACK PROBLEM:";
cout<<"\n\t enter number of objects:";
cin>>n;
for(i=1;i<n;i++)

{
cout<<"\n\t enter profit of objects:";
cin>>p[i];
cout<<"\n\t enter the weight of objects:";
cin>>w[i];
}
cout<<"\n\t Enter the capacity of knapsack:";
cin>>u;
for(i=1;i<n;i++)
{
if(w[i]>u)break;
x[i]=1;
u=u-w[i];
profit=profit+p[i]*x[i];
}
if(i<=n)x[i]=u/w[i];
cout<<"profit:";
for(i=1;i<=n;i++)
cout<<p[i]<<"\n";
cout<<"weight:";
for(i=1;i<n;i++)
cout<<w[i]<<"\n";
cout<<"\n\t Result:";
for(i=1;i<n;i++)

{
cout<<x[i]<<" ";
}
cout<<"\n Maximize profit is:"<<profit;
getch();
}
```

OUTPUT:

KNAPSACK PROBLEM:

Enter number of objects:3

Enter profit of objects:12

Enter the weight of objects:39

Enter profit of objects:44

Enter the weight of objects:55

Enter the capacity of knapsack:66

Profit:
12
44
0

Weight:
39
55

Result:1 0.490909
Maximize profit is:12

Result:

Thus the program is successfully executed and output is verified.

Activate Windows
Go to Settings to activate Windows.

Ex. No : 8	DIVIDE AND CONQUER METHOD.
Date : 2/11/22	

Aim :
To work a program to search for an element in a tree using divide and conquer method.

Algorithm :

Step 1: Start the program.

Step 2: Declare the class prime and data member and number function

Step 3: Declare the variables and function the value

Step 4: Define void element(), function and void spend () function

Step 5: Initializing the variable v_1, v_2, j, w

Step 6: Derived base class into the function

Step 7: Stop the process.

DIVIDE AND CONQUER METHOD

```
#include<iostream.h>
#include<conio.h>
int main()
{

clrscr();
int n,a[30],item,i,j,mid,top,bottom;
cout<<"\n DIVIDE AND CONQUER METHOD";

cout<<"\n ****** *** ******* ******"; cout<<"\n Enter how
many elements you want:"; cin>>n;

cout<<"\n Enter the elements in ascending order:";
for(i=0;i<n;i++)

{
cin>>a[i];
}
cout<<"\n Enter the item to search:";
cin>>item;
bottom=1;

top=n;
do

{
mid=(bottom+top)/2;
if(item<a[mid])
top=mid-1;
else if(item>a[mid])

bottom=mid+1;
}
while(item!=a[mid]&&bottom<=top);

if(item==a[mid])
{
cout<<"\n Binary search successfull";
cin>>item;
cout<<"\n        Found in position:"<<item,mid+1;

}
else
{
cout<<"\n Search failed\nNot found";

cin>>item;
}
return 0;
}
```

OUTPUT:

```
DIVIDE AND CONQUER METHOD
****** *** ******* ******
Enter how many elements you want:12

Enter the 12 elements in ascending order:11
22
33
44
55
66
77

88
99
111
222
333

Enter the item to search:67
Search failed
67Not found
```

Result:

Thus the program is successfully executed and output is verified.

Activate Windows
Go to Settings to activate Windows.

Ex. No : 9	QUEEN ON 8*8 MATRIX
Date : 04/11/22	

Aim :
To write a program to place the 8 queens on an 8*8 matrix, so that no two queens attach.

Algorithm :

Step 1: Start the program.

Step 2: Track the eight queen problem views back tracking solution.

Step 3: Every on the solve dialog that goes from the upper right two lower left.

Step 4: Estimate the number of bnds that will be generated by queen.

Step 5: Place return a boolean value that a true if the 1 queen can be place

Step 6: Stop the program.

UEEN ON 8*8 MATRIX

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#include<iostream.h>

static int t[10]={-1};
void queens(int i);
int empty(int i);

void print_solution();
int main()
{
clrscr();
cout<<"\n\t 8 QUEENS PROBLEM";

cout<<"\n\t*****************";
queens(1);
print_solution();

return(0);
}
void queens(int i)
{
for(t[i]=1;t[i]<=8;t[i]++)

{
if(empty(i))
{
if(i==8)

{
print_solution();
exit(0);
}
else
{

queens(i+1);
}

}
}
}
int empty(int i)
{

int j;
j=1;
while(t[i]!=t[j] && abs(t[i]-t[j])!=(i-j) && j<8)

j++;
```

```
return((i==j)?1:0);
}
void print_solution()
{

int i;
for(i=1;i<=8;i++)
cout<<"\nt["<<i<<"]="<<t[i];

getch();
}
```

OUTPUT:

8QUEENS PROBLEM

****

t[1]=[1]

t[2]=[5]

t[3]=[8]

t[4]=[6]

t[5]=[3]

t[6]=[7]

t[7]=[2]

t[8]=[4]

Result:

Thus the program is executed and output is verified.

Ex. No : 10	VIRTUAL FUNCTION.
Date : 07/11/22	

Aim :

To create a program to perform virtual function

Algorithm :

Step 1 : Start the program

Step 2: Declare base class.

Step 3: Declare and define the virtual function show()

Step 4: Declare and define the function display()

Step 5: Create and define class from the baseclass

Step 6: Declare and defined the function display() and show()

Step 7: Create the base class object and pointer variable

Step 8: Call the function display() and show() using the base class objects and pointer.

Step 9: Create the derived class object and call the function display() and show() using the derived class object pointer.

Step 10: Stop the program.

VIRTUAL FUNCTION

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
void line ();
class shap
{
public:
virtual void call_area()
{
}

virtual void call_peri()
{
}
};
class square:public shap
{
int a;
public:
square(int x)
{
a=x;
}
void call_area();
void call_peri();
};
class rect:public shap
{
int l,b;
public:
rect(int x,int y)
{
l=x;
b=y;
}

void call_area();
void call_peri();
};
class tri:public shap
{
int b,h,c;
public:
tri(int x,int y,int z)
```

```
b=x;
h=y;
c=z;
}
void call_area();
void call_peri();
};
void square::call_area()
{
cout<<endl<<"\n area of the square:"<<a*a;
}
void square::call_peri()
{
cout<<endl<<"\n perimeter of the square:"<<4*a;
}
void rect::call_area()
{
cout<<endl<<"\n area of rectangle:"<<(l*b);
}
void rect::call_peri()
{
cout<<endl<<"\n perimeter of the rectanle:"<<2*(l+b);
}
void tri::call_peri()
{
cout<<endl<<"\n perimeter of the triangle:"<<b+h+c;
}
void tri::call_area()
{
float s;
s=(b+h+c)/2;
s=sqrt(s*(s-b)*(s-h)*(s-c));
cout<<endl<<"\n area of the triangle:"<<s;
}
void main()
{
int asqr,lrec,brec,btri,htri,ctri;

clrscr();
cout<<"\n\n\t find area and perimeter:"<<endl;
line();
cout<<endl<<"enter the size of the square:";
cin>>asqr;

cout<<endl<<"\n enter the breadth and length of the rectangle:";
cin>>brec>>lrec;
```

```
cout<<endl<<"\n enter three sides of the
triangle:"; cin>>btri>>htri>>ctri;
 line();
cout<<"request";
square s(asqr);
rect r(brec,lrec);
tri t(btri,htri,ctri);
shap*sp;
sp=&s;
sp->call_area();
sp->call_peri();
sp=&t;
sp->call_area();
sp->call_peri();
line();

getch();
}
void line()
{
int i;
cout<<endl;

for(i=1;i<=80;i++)
cout<<"~";
cout<<endl;
}
```

OUTPUT:

```
Find area and perimeter:
---------------------------------------------------------
Enter the size of the square:25

Enter the breadth and length of the rectangle:12
23
Enter three sides of the triangle:5
6
3
---------------------------------------------------------
Request
Area of the square:625
Perimeter of the square:100
Area of the triangle:7.483315
Perimeter of the triangle:14
---------------------------------------------------------
```

Result:

Thus the program is successfully executed and output is verified.

Ex. No : 11	PARAMETERIZED CONSTRUCTOR.
Date : 9/11/22	

Aim :

To write a c++ program to perform parameterized constructor.

Algorithm :

Step 1: Start the program

Step 2: Declare class or prime with data number & number functn

Step 3: Consider the argument constructor prime or with Integer argument

Step 4: To call function Calculate () and to the following steps

Step 5: Check whether the ic value n increment ble u ans 1

Step 6: If u than display the value

Step 7: Stop the program.

PARAMETERIZED CONSTRUCTOR

```
#include<iostream.h>
#include<conio.h>
class complex
{
float x,y;
public:
complex()
{
}
complex(float a)
{
x=y=a;
}
complex(float real,float image)
{
x=real;
y=image;
}
friend complex sum(complex,complex);
friend void show(complex);
};
complex sum(complex c1,complex c2)
{
```

```
complex c3;
c3.x=c1.x+c2.x;
c3.y=c1.y+c2.y;
return c3;
}
void show(complex c)
{
cout<<c.x<<"+j"<<c.y<<"\n";
}
int main()
{
clrscr();
complex a(2.7,3.5);
complex b(1.6);
complex c;
c=sum(a,b);
cout<<"A=";
show(a);
cout<<"B=";
show(b);
cout<<"c=";
show(c);
complex P,Q,R;
P=complex(2.5,3.9);
Q=complex(1.6,2.5);
```

```
R=sum(P,Q);
cout<<"\n";
cout<<"p=";
show(P);
cout<<"Q=";
show(Q);
cout<<"R=";
show(R);
getch();
}
```

OUTPUT:

A=2.7+j3.5

B=1.6+j1.6

c=4.3+j5.1

p=2.5+j3.9

Q=1.6+j2.5

R=4.1+j6.4

Result:

Thus the program is successfully executed and output is verified.

Ex. No : 12	FRIEND FUNCTION
Date : 11/11/22	

Aim :

To write a program using friend function

Algorithm :

Step 1 : Start the program

Step 2: Declare the class name as base with data member and member function.

Step 3: The function getc() is used to read the 2 inputs from user

Step 4: Declare the friend function mean(), base include the class

Step 5: Outside the class to define the friend function and do the following.

Step 6: Return the mean value 2 as a float

Step 7: Stop the program

FRIEND FUNCTION

```
#include<iostream.h>
#include<conio.h>
class two;
class one
{

int a;
float b;
public:

void get()
{

cout<<"\n ENTER THE INTEGER VALUE FOR a:"; cin>>a;

cout<<"\n ENTER THE FLOAT VALUE FOR b:";
cin>>b;
}
friend void dis(one,two);

void display()
{

cout<<"\n\n FIRST CLASS INTEGER VALUE"<<a<<endl;
cout<<"\n\n FIRST CLASS FLOAT VALUE"<<b<<endl; }

};
class two
{
int c;

float d;
public:
void get()
{

cout<<"\n ENTER THE INTEGER VALUE FOR c:"; cin>>c;

cout<<"\n ENTER THE FLOAT VALUE FOR d:";
cin>>d;

}
friend void dis(one,two);
void display()
{

cout<<"\n\n SECOND CLASS INTEGER VALUE"<<c<<endl;
cout<<"\n\n SECOND CLASS FLOAT VALUE"<<d<<endl; }

};

void dis(one s1,two t1)
{
```

```
cout<<"\n THE FIRST CLASS INTEGER VALUE IS:"<<s1.a<<endl; cout<<"\n
THE FIRST CLASS FLOAT VALUE IS:"<<s1.b<<endl; cout<<"\n THE SECOND
CLASS INTEGER VALUE IS:"<<t1.c<<endl; cout<<"\n THE SECOND CLASS
FLOAT VALUE IS :"<<t1.d<<endl; }

void main()

{
one s1;
two t1;

clrscr();

cout<<"\n ENTER THE FIRST CLASS VALUE"<<endl<<endl; s1.get();

cout<<"\n ENTER THE SECOND CLASS VALUE"<<endl<<endl;
t1.get();

clrscr();
s1.display();
t1.display();

cout<<"\n\n     USING     FRIEND     FUNCTION"<<endl;
cout<<"\n\n ~~~
```

OUTPUT:

ENTER THE FIRST CLASS VALUE
ENTER THE INTEGER VALUE FOR a:2
ENTER THE FLOAT VALUE FOR b:4.8

ENTER THE SECOND CLASS VALUE
ENTER THE INTEGER VALUE FOR c:5
ENTER THE FLOAT VALUE FOR d:9.7

FIRST CLASS INTEGER VALUE2
FIRST CLASS FLOAT VALUE4.8
SECOND CLASS INTEGER VALUE5
SECOND CLASS FLOAT VALUE9.7

USING FRIEND FUNCTION
~~~~~~~~~~~~~~~~~~~~~~

THE FIRST CLASS INTEGER VALUE IS:2
THE FIRST CLASS FLOAT VALUE IS:4.8 THE
SECOND CLASS INTEGER VALUE IS:5 THE
SECOND CLASS FLOAT VALUE IS :9.7

Result:
Thus the program is successfully executed and output is verified.
~~~~~~~~~~~~~~~~~~~~~~

Ex. No : 13	FUNCTION OVERLOADING
Date : 14/11/22	

Aim :

To write a C++ program using function overloading.

Algorithm :

Step 1 : Start the program

Step 2 : Declare the class as function with data number and member function

Step 3 : Read the choice from the user

Step 4 : Choice = 1 then go to the steps

Step 5 : The function areacs to find area of circular one integer argument

Step 6 : Choice = 2 then go to the step 7

Step 7 : the function areacs to find area of rectangle with two integer argument.

Step 8 : Choice = 3 then go to step 9

Step 9 : The function areacs to find area of triangle with three argument two or integer and one as float

Step 10 : Stop the program

FUNCTION OVERLOADING

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
class INT
{
protected:
int a[5][5];
};
class FLOAT
{
protected:
float b[5][5];
};
class ADD:public INT, public FLOAT
{
protected:
float c[5][5],tot,bgh;
int m,n,i,j,agh;
public:
void getdata();
void read(INT);
void read(FLOAT);
void sum(INT);
```

```
void sum(FLOAT);
void totsum();
void matadd(INT,FLOAT);
void disp();
};
void ADD::getdata()
{
cout<<"\n enter the number of rows & columns:";
cin>>n>>m;
}
void ADD::read(INT)
{
cout<<"\n enter the integer value for matrix A:";
for(i=0;i<n;i++)
for(j=0;j<m;j++)
cin>>a[i][j];
}
void ADD::read(FLOAT)
{
cout<<"\n enter the float value matrix B:";
for(i=0;i<n;i++)
for(j=0;j<m;j++)
cin>>b[i][j];
}
void ADD::sum(INT)
```

```
{
agh=0;
for(i=0;i<n;i++)
for(j=0;j<m;j++)
agh=agh+a[i][j];
cout<<"\n the total number of integer value for matrix A:"<<agh<<"\n";
}
void ADD::sum(FLOAT)
{
bgh=0;
for(i=0;i<n;i++)
for(j=0;j<m;j++)
bgh=bgh+b[i][j];
cout<<"\n the total number of float value for matrix B:"<<bgh<<"\n";
}
void ADD::totsum()
{
tot=agh+bgh;
cout<<"\n the total number of integer & float value of matrix(A+B):"<<tot;
}
void ADD::disp()
{
cout<<"\n the value of matrix A:\n";
for(i=0;i<n;i++)
{
```

```
for(j=0;j<m;j++)
{
cout<<"\t"<<a[i][j]<<"";
}
cout<<"\n \n";
}
cout<<"\n the value of matrix B:\n";
for(i=0;i<n;i++)
{
for(j=0;j<m;j++)
{
cout<<"\t"<<b[i][j]<<"";
}
cout<<"\n \n";
}
}
void ADD::matadd(INT,FLOAT)
{
for(i=0;i<n;i++)
for(j=0;j<m;j++)
c[i][j]=a[i][j]+b[i][j];
cout<<"\n Resultant matrix is \n";
for(i=0;i<n;i++)
{
for(j=0;j<m;j++)
```

```
{
cout<<"\t"<<c[i][j]<<"";
}
cout<<"\n \n";
}
}
void main()
{
INT I1;
FLOAT F1;
ADD A1;
clrscr();
cout<<"\n \n \n \n \n";
cout<<setw(24)<<"\t\t matrix addition using function overloading\n";
cout<<setw(25)<<"\t \t ~~~
```

OUTPUT:

```
Enter the integer value for matrix A : 4
5
6
7
8
9
2
3
5

Enter the float value for matrix B : 3.2
2.4
6.7
7.8
3.5
4.6
7.1
5.3
2.4
The value of matrix A:
    4    5    6

    7    8    9

    2    3    5
The value of matrix B:
    3.2   2.4   6.7

    7.8   3.5   4.6

    7.1   5.3   2.4
Resultant matrix is:
    7.2   7.4   12.7

    14.8  11.5  13.6

    9.1   8.3   7.4

the total number of integer value for matrix A:49

the total number of float value for matrix B:43

the total number of integer and float value of matrix(A+B):92
```

Result:

Thus the program is successfully executed and output is verified

Ex. No : 14	SINGLE INHERITANCE.
Date : 15/11/22	

Aim :

To write a program to perform single Inheritance

Algorithm :

Step 1: Start the program

Step 2: Declare and call dind class B

Step 3: Declare the variable and number function data function.

Step 4: Find the derived class object and declare the variable

Step 5: Create a mult() function and display function

Step 6: Display the result

SINGLE INHERITANCE

```
#include<iostream.h>
#include<conio.h>
class B
{
int a;
public:
int b;
void get_ab();
int get_a(void);
void show_a(void);
};
class D:public B
{
int c;
public:
void mul(void);
void display(void);
};
void B::get_ab()
{
a=5;
b=10;
}
```

```
int B::get_a(void)
{
return a;
}
void B::show_a(void)
{
cout<<"a="<<a<<"\n";
}
void D::mul()
{
c=b*get_a();
}
void D::display()
{
cout<<"a="<<get_a()<<"\n";
cout<<"b="<<b<<"\n";
cout<<"c="<<c<<"\n";
}
void main()
{
clrscr();
D d;
d.get_ab();
d.mul();
d.show_a();
```

```
d.display();
d.b=20;
d.mul();
d.display();
getch();
}
```

OUTPUT:

a=5

a=5

b=10

c=50

a=5

b=20

c=100

Result:
Thus the program is successfully executed and output is verified

Ex. No : 15	EMPLOYEE DETAILS
Date : 16/11/22.	

Aim :

To write a program to perform employee details.

Algorithm :

Step 1: Start the program

Step 2: Declare the base class employee.

Step 3: Define and declare function get() to employee details.

Step 4: Declare and derived class salary.

Step 5: Declare and define function get() to salary details

Step 6: Define function calculate() to find displtonetpay

Step 7: Define the function

Step 8: Create derived class object

Step 9: Read the employee number

Step 10 Call the display() function

Step 11: Stop the program

EMPLOYEE DETAILS

```
#include<iostream.h>
#include<conio.h>
Class emp
{
Public:
Char name[20,des[20];
Void get()
{
Cout<"Enter the employee number: " ;
Cin>>eno;
Cout<<"Enter the employee name: ";
Cin>>name;
Cout<<"Enter the designation:";
Cin>>des;
}
};
Class salary:public emp
{
Float bp,hra,da,pf,np;
Public:
Void get()
{
Cout<<"Entyer the basic pay:";
Cin>.bp;
Cout<<"Enter the human resource allowance:";
Cin>.da;
Cout<<"Enter the profitability fund:";
Cin>>pf;
}
Void calculate()
{
Np=bp+hra+da-pf;
}
Cout<<eno"\t"<<name<<\t"<<des<<\t"<<hra<<"\t"<<da<<"\t<<"pf<<"\t"<<np<<"\t";
}
};
Void main()
{
Int I,n;
Char ch;
Salary s[10];
Clrscr();
Cout<<"Enter the number of employee:";
Cin>>n;
For(i=0;i<n;i++)
{
S[i].get();
s[i].calculate();
}
```

```
Cout<<"\n e-no\t e-name\t des\t bp\t hra\t da\t pf\t np\n";
For(i=0;i<n;i++)
{
S[i].display();
}
Getch();
}
```

OUTPUT:

Enter the number of employee:1

Enter the employee no:150

Enter the employee name:Ram

Enter thedesignation:Mnager

Enter the basic pay:5000

Enter the HR allowance:1000

Enter the dearness allowance:500

Enter the profitability fund:300

Result:

Thus the program is successfully executed and output is verified.