No:1 Factorial of a given number

Algorithm

Step 1: start
Step 2: read the number n
Step 3: call fact(n)
Step 4: print fact f
Step 5: stop

AIM: Write a C program to find factorial of a given number.

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

float fact(int);
void main()
{
float f;
int n;
clrscr();
printf("Enter the number: \n");
scanf("%d",&n);
f=fact(n);
printf("factorial is:  %f",f);
getch();
}

float fact(int n)
{
int i;
```

```
float f=1;
if(n==0)
f=1;

else
{
for(i=1;i<=n;i++)
f=f*i;

return f;
}
}
```

No:2 Fibonacci series

Algorithm

Step 1: start
Step 2: declare variables f1,f2,f3 & i
Step 3: initialize the variables f1=0,f2=1,i=2
Step 4: enter the number of terms of series to be printed
Step 5: print first two terms of series
Step 6: use loop for the following steps
f3=f2+f1
f1=f2
f2=f3
increase value of i each time by 1
print the value of f3
Step 7: stop

AIM: Write a C program to display the Fibonacci series

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
int n,f1=0,f2=1,f3,i=2;
clrscr();
printf("enter the limit");
scanf("%d",&n);
printf("\nthe fibonacci series\n");
printf("%d\n%d\n",f1,f2);
while(i<n)
```

```
{
f3=f2+f1;
f1=f2;
f2=f3;
printf("%d\n",f3);
i=i+1;
}
getch();
}
```

No:3 Sort

Algorithm

Step 1: start
Step 2: read 10 inputs for array a
Step 3: read for i=1 to N by 1
Read for j=1 to 10-1 by 1
if a[j]>a[j+1] then:
temp=a[j]
a[j]=a[j+1]
a[j+1]=temp
end of if structure
end of inner for loop
end of outer for loop
Step 4: print array a[i]
Step 5: exit

AIM: Write a C program to sort given set of numbers.

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

int main()
{

int a[10],i,j,temp=0;
clrscr();
printf("Enter all the 10 numbers: \n");
for(i=0;i<10;i++)
```

```
scanf("%d",&a[i]);
for(i=0;i<10;i++)
{
for(j=0;j<9;j++)
{

if(a[j]>a[j+1])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
}
printf("The ordered array is:  \n");
for(j=0;j<10;j++)
printf("%d\t",a[j]);
getch();
return 0;
}
```

No:4 Palindrome

Algorithm

Step 1: Start

Step 2: Input a string str[30];

Step 3: Declare two pointers *p & *t

Step 4: Print “enter the string”

Step 5: Read the string and store it into string ‘str’

Step 6: Use for loop for comparison of characters use; in first for loop so p value is going to end of string (NULL)

Step 7: Second for loop t = str& p is decrementing (p--), repeat till all checking starting & ending character by character

Step 8: if this condition unequal goes to break

Step 9: if (t>p) print “string is palindrome”
else print “string is not palindrome”

Step 10: Stop

AIM: Write a C program to check whether the given string is a palindrome or not using pointers.

```
#include<stdio.h>
#include<conio.h>
int main()
{
char str[30];
char *p,*t;
clrscr();
printf("Enter any string :");
gets(str);
for(p=str;*p!=NULL;p++);
for(t=str;p--;p>=t)
{
if(*p==*t)
{
p--;
t++;
}
else
break;
}
if(t>p)
printf("\n string is palindrome");
else
printf("\n string is not palindrome");
getch();
return 0;
}
```

No:5 File creation and display

Algorithm

Step 1: Start

Step 2: Declare the file pointer fp

Step 3: Write the file name

Step 4: Read the first file to read from

Step 5: Open file in read mode

Step 6: If any error occur exit

Step 7: Read character by character from the file and
display until end of file reached

Step 8: Count the number of lines in the file and display

Step 9: Stop

AIM: Write a C program to create a file and to display the contents of that file with line numbers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
FILE *fp;
char ch;
char source[67];
int count=1;
```

```
clrscr();
puts("Enter the file name:  ");
gets(source);
fp=fopen(source,"r");
if(fp==NULL)
{
puts("unable to open the file:  ");
getch();
exit();
}
printf("file name:%s",source);
printf("\n line:-%d\t",count);
while((ch=getc(fp))!=EOF)
{
if(ch=='\n')
{
count++;
printf("\n line:-%d\t",count);
}
else


{
printf("%c",ch);
}
}
printf("\n press any key...");
getch();
fclose(fp);
}
```

No:6 Merging of two files

Algorithm

Step 1: Read the first file name to read from

Step 2: Read the second file name to read from

Step 3: Read the third file name to store the contents of the two files

Step 4: Open file 1 and file 2 in read mode

Step 5: If any of the file doesn't open exit

Step 6: Open file 3 in write mode

Step 7: If any error occur exit

Step 8: Read character by character from file 1 and write to file 3 until end of file is reached

Step 9: Read character by character from file 2 and write to file 3 until end of file is reached

Step 10: Print the merging is successful

Step 11: Stop

AIM: Write a C program to merge two files into a single file.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
void main()
{

FILE *fs1,*fs2,*ft;
char ch,file1[20],file2[20],file3[20];
clrscr();

printf("Enter name of first file\n");
gets(file1);

printf("Enter name of second file\n");
gets(file2);

printf("Enter name of file which will store contents of two files\n");
gets(file3);

fs1=fopen(file1,"r");
fs2=fopen(file2,"r");

if(fs1==NULL || fs2==NULL)
{
perror("Error");


printf("Press any key to exit...\n");


getch();

exit(EXIT_FAILURE);
}
```

```
ft=fopen(file3,"w");

if(ft==NULL)
{
perror("Error");
printf("Press any key to exit...\n");
exit(EXIT_FAILURE);
}

while( (ch=fgetc(fs1) )!=EOF)
fputc(ch,ft);

while( (ch=fgetc(fs2) )!=EOF)
fputc(ch,ft);

printf("Two files were merged into %s file successfully.\n",file3);

fclose(fs1);
fclose(fs2);
fclose(ft);

getch();
}
```

No:7 Stack

Algorithm

Step 1: Include all the header files which are used in the program & define a consist size with specific value

Step 2: Declare all the functions used in stack implementation

Step 3: Create a one dimensional array with fixed size (int item [size])

Step 4: Define a integer variable top and initialize with '-1' top= -1

Step 5: In main method display menu with list of operations and make suitable function calls to perform operation selected by the user on the stack

PUSH(): Inserting value into stack.

Step 1: Check whether stack is full (top==size-1)

Step 2: If it is full, then display "stack overflow"

Step 3: If it is not full, then increment top value by one (top++) and set stack [top] to value

(item[top] = elem).

POP() : Delete a value from the stack.

Step 1: Check whether stack is empty if (top<0)

Step 2: If it is empty, then display "stack is underflow and terminate function"

Step 3: If it is not empty, then delete item and display "popped element is" and decrement top value by one (top--)

DISPLAY() : Display the elements of a stack

Step 1: Check whether stack is empty if (top<0)

Step 2: If it is empty, then display "stack is empty" and terminate the function

Step 3: If it is not empty, then define a variable 'i' and initialize with top. Display item[i] value and decrement I value by one (i--) repeat above step until I value become '0'

Step 4: Stop

AIM: Write a C++ program to create a class to implement the data structure STACK.Write a constructor to initialize the TOP of the STACK.Write a member function PUSH() to insert an element and member function POP() to delete an element check for overflow and underflow conditions.

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<stdlib.h>
const int size=10;
class stack
{
private:
int top;
int item[size];
public:
stack()
{
top=-1;
}
~stack()
{
delete item;
}
void create();
void push();
void pop();
void display();
};

void stack::create()
{
int n,i;
```

```
cout<<"Enter how many elements"<<endl;
cin>>n;
if(n>=size)
cout<<"stack can hold only "<<size-1<<"element"<<endl;
else
{
cout<<"Enter the element"<<endl;
for(i=0;i<n;i++)
cin>>item[++top];
}
}
void stack::push()
{
int elem;
if(top==size-1)
cout<<"stack overflow"<<endl;
else
{
cout<<"Enter the element to be pushed"<<endl;
cin>>elem;
top++;
item[top]=elem;

}
}
void stack::pop()
{
if(top<0)
cout<<"stack underflow"<<endl;
else
{
cout<<"poped element is"<<endl;
top--;
}
}

void stack::display()
```

```
{
if(top<0)
cout<<"stack is empty"<<endl;
else
{
cout<<"element of stack are "<<endl;
for(int i=top;i>=0;i--)
cout<<item[i]<<endl;
}
}
void main()
{
clrscr();
int choice;
char ch;
stack s;
do
{
cout<<"MAIN MENU"<<endl;
cout<<"==========="<<endl;

cout<<"1.create a stack"<<endl;
cout<<"2.push"<<endl;
cout<<"3.pop"<<endl;
cout<<"4.display"<<endl;
cout<<"5.exit"<<endl;
cout<<"enter your choice"<<endl;
cout<<"==========="<<endl;
cin>>choice;
switch(choice)
{
case 1:s.create();
break;
case 2:s.push();
break;
case 3:s.pop();
break;
```

```
case 4:s.display();
break;
case 5:exit(0);
break;
default:cout<<"invalid choice"<<endl;
}
cout<<"to continue: press 'y' "<<endl;
cin>>ch;
}
while(ch=='y'||ch=='y');
getch();
}
```

No:8 Program using inline function,constructor & destructor

Algorithm

Step 1: Start

Step 2: Declare class ”digit”

Step 3: Declare the variable ‘a’.’sum’

Step 4: Initialize the constructor with parameter “s”

Step 5: a=s

Step 6: Initialize the destructor

Step 7: Defining the inline function ‘calc ()’.

Step 8: Read: take input ‘a’

Step 9: Initialize ‘sum=0’

Step 10: Repeat while (a>0)

Set x = a%10

a = a%10

sum =sum+ x

Step 11: Print: sum of digits is ‘sum’

Step 12: If (sum/10) not equal to 10

y= sum%10

sum= sum/10

sum= sum+ y

Step 13: Print: “sum of digits” is ‘sum’

Step 14: Print: “inline function ”

Step 15: call the inline function

Step 16: Stop

AIM: Write a C++ program to read an integer number and find the sum of all the digits until it reduces to a single digit using constructors,destructors and inline member function.

```
#include<iostream.h>
#include<conio.h>
class digit
{
int a,sum;
public:
digit(int s)
{
a=s;
cout<<"\n\n value is\t"<<s;
}
~digit() {}
inline void calc()
{
sum=0;
while(a>0)
{
int x=a%10;
a=a/10;
sum=sum+x;
}
cout<<"\n\n sum of digit\t"<<sum;
if((sum/10)!=10)
```

```
{
int y=sum%10;
sum=sum/10;
sum=sum+y;
}
cout<<"\n\n sum of digits \t"<<sum;
}
};
void main()
{
clrscr();
cout<<"inline function\n";
cout<<"============\n";
digit d(789);
d.calc();
getch();
}
```

No:9 Operator overloading

Algorithm

Step 1: Start

Step 2: Create a class 'FLOAT' with a and b as
data members and get data () to read values and
display () to display values

Step 3: Create 3 objects d1, d2, and d3

Step 4: Call get data () using object d1, and d2
separately and read values

Step 5: Perform arithmetic operations by calling four
overloading functions separately and display
values separately using object d3

Step 6: Stop.

AIM: Write a C++ program to create a class FLOAT that contains one float data member.Overload all the four arithmetic operators so that they operate on the object FLOAT.

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
class FLOAT
{
float a,b;
public:
void getdata();
void display();
FLOAT operator+(FLOAT);
FLOAT operator-(FLOAT);
FLOAT operator*(FLOAT);
FLOAT operator/(FLOAT);
};
void FLOAT::getdata()
{
cout<<"enter the values of a and b:";
cin>>a>>b;
}
void FLOAT::display()
{
cout<<"\na="<<a<<"\nb="<<b;
}
FLOAT FLOAT::operator+(FLOAT d1)
{
FLOAT d2;
d2.a=a+d1.a;
d2.b=b+d1.b;.
return d2;
}
FLOAT FLOAT::operator-(FLOAT d1)
```

```
{
FLOAT d2;
d2.a=a-d1.a;
d2.b=b-d1.b;
return d2;
}
FLOAT FLOAT::operator*(FLOAT d1)
{
FLOAT d2;
d2.a=a*d1.a;
d2.b=b*d1.b;
return d2;
}
FLOAT FLOAT::operator/(FLOAT d1)
{
FLOAT d2;
d2.a=a/d1.a;
d2.b=b/d1.b;
return d2;
}
void main()
{
clrscr();
FLOAT d1,d2,d3;
cout<<"enter first object\n";
d1.getdata();
cout<<"enter second object\n";
d2.getdata();
d3=d1+d2;
d3.display();
d3=d1-d2;
d3.display();
d3=d1*d2;
d3.display();
d3=d1/d2;
```

```
d3.display();
getch();
}
```

No:10 String Concatenation

Algorithm

Step 1: Start

Step 2: Create class string with ‘s’ string data members
‘st1’, ‘st2’ and st and 3 member functions
Read () for reading, concate() for concatenation
cmp () for comparing two strings

Step 3: Create object x for class string, call x. Read () to
read 2 string call x. Concate (), to copy string st1
to st2 and concatenate string st and string st2
and store string in st

Step 4: Print string in ‘st’
Call x. cmpr (); to compare two string using
Condition
strcmp (st1, st2 ==0)
If true print it is equal
else
Print not equal

Step 5: Stop

AIM: Write a C++ program to create a class STRING.Write a member function to initialize,get and display strings.Overload the operator "+" to concatenate two strings, "==" to compare two strings.

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
class string
{
public:
char st1[15];
char st2[15];
char st[30];
void read();
void concat();
void cmpr();
};
void string::read()
{
cout<<"\n enter first string:  ";
cin>>st1;
cout<<"\n enter second string:  ";
cin>>st2;
}
void string::concat()
{
strcpy(st,st1);
strcat(st,"");
strcat(st,st2);
cout<<"\n after concatenate:  "<<"\n"<<st;
}
void string::cmpr()
{
if(strcmp(st1,st2)==0)
cout<<"\n they are equal";
```

```
else
cout<<"\n they are not equal";
}
void main()
{
string x;
clrscr();
x.read();
x.concat();
x.cmpr();
getch();
}
```

No:11 Friend function between two classes

Algorithm

Step 1: Start

Step 2: Create class ‘first’ with data member a and b and member function get () to access the value of a and b

Step 3 : Create a member function display to display a and b

Step 4: Create objects s1 for first class s2 for second class

Step 5: Create class ‘second’ with c and d as data members and member functions get1 () to read values and display1 to display values

Step 6: Create a friend function named value with arguments s1 and s2 which are object of both classes

Step 7: Print the value of a, b, c, d with passed object arguments s1 and s2

Step 8: Stop.

AIM: Write a C++ program to create two classes each class consists of two private variables,a integer and a flaot variable.Write member function to get and display them.Write a FRIEND function common to both classes,which take the object of above two classes as arguments and the integer and float values of both objects separately and display the result.

```
#include<iostream.h>
#include<conio.h>
class second;
class first
{
private:
int a;
float b;
public:
void get()
{
cout<<"\n Enter the values of integer a and floating b:  ";
cin>>a>>b;
}
void display()
{
cout<<"\n value of a and b is =  "<<a<<"\n"<<b;
}
friend void value(first s1,second s2);
};
class second
{
private:
int c;
float  d;
public:
```

```
void get1()
{
cout<<"\n Enter the value of integer c and floating d:  ";
cin>>c>>d;
}
void display1()
{
cout<<"\n value of c and d is=  "<<c<<"\n"<<d;
}
friend void value(first s1,second s2);
};
void value(first s1,second s2)
{
cout<<"\n\n\n\n\n\n value of a and b using friend function is=
        "<<s1.a<<"\n"<<s1.b;
cout<<"\n value of c and d using friend function is=
        "<<s2.c<<"\n"<<s2.d;
}
void main()
{
clrscr();
first s1;
second s2;
s1.get();
s1.display();
s2.get1();
s2.display1();
value(s1,s2);
getch();
}
```

No:12 Function overloading

Algorithm

Step 1: Start

Step 2: Enter the row and column of the matrix

Step 3: Enter the elements of the ‘mat1’ matrix

Step 4: Enter the elements of the ‘mat2’ matrix

Step 5: Print the ‘mat1’ matrix

Step 6: Print the ‘mat2’ matrix

Step 7: Set a loop up to the row

Step 8: Set a inner loop up to the column

Step 9: Add the elements of ‘mat1’ and ‘mat2’ in column wise and store the result in ‘mat3’ matrix

Step 10: After the execution of the two loops print the value of ‘mat3’ matrix

Step 11: Stop

AIM: Write a C++ program using Function Overloading to read two matrices of different Data Types such as integers and floating point numbers.Find out the sum of the above two matrices separately and display the sum of these arrays individually.

```
#include<iostream.h>
#include<conio.h>
void main()
{
clrscr();
int mat1[3][3],i,j;
float mat2[3][3],mat3[3][3];
cout<<"enter matrix 1 element :\n";
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
cin>>mat1[i][j];
}
}
cout<<"enter matrix 2 element :\n";
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
cin>>mat2[i][j];
}
}
cout<<"adding the two matrix to form the third
matrix.....\n";

for(i=0;i<3;i++)
{
```

```
for(j=0;j<3;j++)
{
mat3[i][j]=mat1[i][j]+mat2[i][j];
}
}
cout<<"the two matrix added successfully....!!";
cout<<"the new matrix will be :\n";
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
cout<<mat3[i][j]<<" ";
}
cout<<"\n";
}
getch();
}
```