INDEX

1. a) Write a C++ program to find the sum of individual digits of a positive integer.

```
#include<iostream>
using namespace std;
int sum_of_digits(int n){
int digit,sum=0;
while(n!=0){
digit=n%10;
sum=sum+digit;
n=n/10;
}
return sum;
}
int main(){
int number,digits_sum;
cout<<"Enter Positive integer within the range:";
cin>>number;
digits_sum = sum_of_digits(number);
cout<<"sum of digts of "<<number<<" is "<<digits_sum;
return 0;
}
```

Output:

```
Enter Positive integer within the range:125
sum of digts of 125 is 8
```

b) Write a C++ Program to generate first n terms of Fibonacci sequence.

```
#include<iostream>
using namespace std;
void fib(int n){
int n0,n1,next,count=0;
n0=0;
n1=1;
while(count<n){
cout<<n0<<"\t";
count++;
next=n0+n1;
n0=n1;
n1=next;
}
}
int main(){
int terms;
cout<<"Enter How many terms to be printed:";
cin>>terms;
fib(terms);
return 0;
}
```

Output:

```
Enter How many terms to be printed:5
0       1       1       2       3
```

2. a) Write a C++ program to generate all the prime numbers between 1 and n, where n is a value supplied by the user.

```
#include<iostream>
using namespace std;
void prime(int n){
int factors;
cout<<"prime numbers are... ";
for(int i=2;i<=n;i++){
factors=0;
for(int j=1;j<=i;j++){
if(i%j==0)
factors=factors+1;
}
if(factors<=2)
cout<<i<<" ";
}
}
int main(){
int n;
cout<<"Enter a integer value:";
cin>>n;
prime(n);
return 0;
}
```

Output:

```
Enter a integer value:5
prime numbers are... 2 3 5
```

b) Write a C++ program to find both the largest and smallest number in a list of integers.

```
#include<iostream>
using namespace std;
int main(){
int a[50],i,n,small,large;
cout<<"Enter The Array Size:";
cin>>n;
cout<<"ENTER ELEMENTS OF ARRAY: ";
for(i=0;i<n;i++)
cin>>a[i];
small=a[0];
large=a[0];
for(i=0;i<n;i++){
if(a[i]<small)
small=a[i];
if(a[i]>large)
large=a[i];
}
cout<<"largest value is: "<<large<<endl;
cout<<"smallest value is: "<<small<<endl;
return 0;
}
```

Output:

```
Enter The Array Size:5
ENTER ELEMENTS OF ARRAY: 5 4 3 1 2
largest value is: 5
smallest value is: 1
```

3. a) Write a C++ program to sort a list of numbers in ascending order.

```
#include <iostream>
using namespace std;
int main(){
int arr[100];
int size,i,j,temp;ui
cout<<"Enter the size of an array: ";
cin>>size;
cout<<"Enter the elements of an array: ";
for(i=0; i<size; i++){
cin>>arr[i];
}
for(i=0; i<size; i++){
for(j=i+1;j<size; j++){
if(arr[j]<arr[i]) {
temp = arr[i];
arr[i] = arr[j];
arr[j] = temp;
}
} }
cout<<"Elements of an array in sorted order :";
for(i=0;i<size;i++){
cout<<arr[i]<<" ";
}
return 0;
}
```

Output:

```
Enter the size of an array: 5
Enter the elements of an array: 5 8 7 1 2
Elements of an array in sorted order :1 2 5 7 8
Process returned 0 (0x0)   execution time : 5.173 s
Press any key to continue.
```

b) Write a Program to illustrate New and Delete Keywords for dynamic memory allocation.

```
#include <iostream>
using namespace std;
int main(){
int* p1,*p2,sum;
p1 = new int;
p2 = new int;
cout<<"Enter the first number: ";
cin>>*p1;
cout<<"Enter the Second number: ";
cin>>*p2;
sum  = *p1+*p2;
cout<<"Sum of value are: "<<sum<<endl;
delete p1;
delete p2;
return 0;
}
```

Output:

```
Enter the first number: 10
Enter the Second number: 20
Sum of value are: 30
```

4. a) Write a program Illustrating Class Declarations, Definition, and Accessing Class Members.

```
#include<iostream>
using namespace std;
class simple{
private: int a;
char b;
float c;
public:
void get_data(){
cout<<"Enter an integer value:";
cin>>a;
cout<<"Enter a character:";
cin>>b;
cout<<"Enter a float value:";
cin>>c;
}
void print_data(){
cout<<"\nValues read from keyboard are\n";
cout<<"Integer value:"<<a<<endl;
cout<<"character is :"<<b<<endl;
cout<<"float value is :"<<c<<endl;
}
};
int main(){
simple s;
s.get_data();
s.print_data();
}
```

Output:

```
Enter an integer value:10
Enter a character:D
Enter a float value:0.5

Values read from keyboard are
Integer value:10
character is :D
float value is :0.5
```

b) Program to illustrate default constructor, parameterized constructor and copy constructors.

```
#include <iostream>
using namespace std;
class code{
int id;
int count;
public: code(){
cout << "Default constructor called\n";
id = 0;
cout << "id=" << id << endl;
}
code(int a) {
cout << "Parameterized constructor called\n";
id = a;
cout << "id=" << id << endl;
}
code(code& x) {
cout << "copy constructor called\n";
id = x.id;
cout << "id=" << id << endl;
}
~code(){
cout << "Object Destroyed" ;
cout << " id=" << id << endl;
}
};
int main(){
code d;
code a(5);
code b=a;
return 0;
}
```

Output:

```
Default constructor called
id=0
Parameterized constructor called
id=5
copy constructor called
id=5
Object Destroyed id=5
Object Destroyed id=5
Object Destroyed id=0
```

c) Write a Program to Implement a Class STUDENT having Following Members:

Member	Description
sname	Name of the student
Marks	array Marks of the student
total	Total marks obtained
Tmax	Total maximum marks

Member functions	Member Description
assign ()	Assign Initial Values
compute ()	to Compute Total, Average
display ()	to Display the Data.

```
#include<iostream>
using namespace std;
class student{
char sname[50];
float marks[6];
float total;
float max_marks;
public: void assign();
void compute();
void display();
};
void student::assign(){
cout<<endl<<"Enter Student Name :";
cin>>sname;
for(int i=0;i<6;i++){
cout<<"Enter marks of subject :"<<i+1<<" : ";
cin>>marks[i];
}
cout<<"Enter Maximum total marks :";
cin>>max_marks;
}
void student::compute(){
total=0;
for(int i=0;i<6;i++)
total+=marks[i];
}
void student::display(){
cout<<"Student Name:"<<sname<<endl;
```

```
cout<<"Marks are\n";
for(int i=0;i<6;i++)
cout<<"Subject "<<i+1<<": "<<marks[i]<<endl;
cout<<" -------------------\n";
cout<<"Total :"<<total<<endl;
cout<<" -------------------\n";
float per;
per=(total/max_marks)*100;
cout<<"Percentage:"<<per;
}
int main(){
student obj;
obj.assign();
obj.compute();
obj.display();
return 0;
}
```

Output:

```
Enter Student Name :Darshan
Enter marks of subject :1 : 20
Enter marks of subject :2 : 40
Enter marks of subject :3 : 50
Enter marks of subject :4 : 35
Enter marks of subject :5 : 25
Enter marks of subject :6 : 45
Enter Maximum total marks :300
Student Name:Darshan
Marks are
Subject 1: 20
Subject 2: 40
Subject 3: 50
Subject 4: 35
Subject 5: 25
Subject 6: 45
 -------------------
Total :215
 -------------------
Percentage:71.6667
```

5. a) Write a Program to Demonstrate the i) Operator Overloading. ii) Function Overloading.

i) Operator Overloading

```
#include<iostream>
using namespace std;
class Complex {
private: int real, imag;
public: Complex(int r = 0, int i = 0){
real = r;
imag = i;
}
Complex operator + (Complex obj){
Complex res;
res.real = real + obj.real;
res.imag = imag + obj.imag;
return res;
}
void print(){
cout << real << " + i" << imag << '\n';
}
};
int main(){
Complex c1(10, 5), c2(2, 4);
Complex c3 = c1 + c2;
c3.print();
}
```

Output:

```
12 + i9
```

ii) Function Overloading

```
#include <iostream>
using namespace std;
void print(int i){
cout<<"Here is int "<<i<<endl;
}
void print(double f){
cout<<"Here is float "<<f<<endl;
}
void print(char const *c){
cout<<"Here is char "<<c<<endl;
}
int main(){
print(10);
print(10.10);
print("Ten");
return 0;
}
```

Output:

```
Here is int 10
Here is float 10.1
Here is char Ten
```

b) Write a Program to Demonstrate Friend Function and Friend Class.

```
#include <iostream>
using namespace std;
class ClassB;
class ClassA {
private: int numA;
friend class ClassB;
public: ClassA(){
numA = 12;
}
};
class ClassB {
private: int numB,sum;
public:  ClassB(){
numB=5;
sum=0;
}
void add(){
ClassA objectA;
cout<<"NumA = "<<objectA.numA<<endl;
cout<<"NumB = "<<numB<<endl;
sum= objectA.numA + numB;
}
friend int sum(ClassB);
};
int sum(ClassB b){
cout<<"Sum of Number is: "<<b.sum;
}
int main(){
ClassB objectB;
objectB.add();
sum(objectB);
return 0;
}
```

Output:

```
NumA = 12
NumB = 5
Sum of Number is: 17
```

6. a) Write a Program to Access Members of a STUDENT Class Using Pointer to Object Members.

```
#include <iostream>
using namespace std;
class Student{
private:
int Regno;
char name[20];
public: Student(){
Regno=0;
};
void inputRegno(){
cout<<"Enter the name: ";
cin>>name;
cout<<"Enter an Register number: ";
cin>>Regno;
}
void displayRegno(){
cout<<"Name is : "<<name<<endl;
cout<<"Register Number is : "<<Regno<<endl;
}
};
int main(){
Student S;
Student *ptr;
ptr = new Student; //creating & assigning memory
ptr->inputRegno();
ptr->displayRegno();
return 0;
}
```

Output:

```
Enter the name: darshan
Enter an Register number: 201308
Name is : darshan
Register Number is : 201308
```

b) Write a Program to Generate Fibonacci Series use Constructor to Initialize the Data Members.

```
#include <iostream>
using namespace std;
class fibonacci{
int n1,n2;
public:
fibonacci(){
n1 = 0; n2 = 1;
}
void series(int n){
int i,next;
cout << n1 << " " << n2 << " " ;
for(i=1; i <= n-2; i++){
next = n1 + n2;
cout << next << " " ;
n1 = n2;
n2 = next;
}
}
};
int main(){
fibonacci fib;
int n;
cout << "FIBONACCI SERIES " << endl ;
cout << "How many numbers do you want ? ";
cin >> n;
fib.series(n);
}
```

Output:

```
FIBONACCI SERIES
How many numbers do you want ? 10
0 1 1 2 3 5 8 13 21 34
```

7. Write a C++ program to implement the matrix ADT using a class. The operations supported by this ADT are:

a) Reading a matrix. b) Addition of matrices. c) Printing a matrix. d) Subtraction of matrices. e) Multiplication of matrices.

```
#include<iostream>
#include<iomanip>
using namespace std;
class matrix{
protected:  int i,j,a[10][10],b[10][10],c[10][10];
int m1,n1,m2,n2;
public:  virtual void read()=0;
virtual void display()=0;
virtual void sum()=0;
virtual void sub()=0;
virtual void mult()=0;
};
class result:public matrix{
public: void read();
void sum();
void sub();
void mult();
void display();
};
void result :: read(){
cout<<"\n enter the order of matrix A: ";
cin>>m1>>n1;
cout<<"\n enter the elements of matrix A: ";
for(i=0;i<m1;i++){
for(j=0;j<n1;j++){
cin>>a[i][j];
}
}
cout<<"\n enter the order of matrix B: ";
cin>>m2>>n2;
cout<<"\n enter the elemnts of matrix B: ";
for(i=0;i<m2;i++){
for(j=0;j<n2;j++){
cin>>b[i][j];
}
}
}
void result :: display(){
for(i=0;i<m1;i++){
for(j=0;j<n1;j++){
```

```
cout.width(3);
cout<<c[i][j];
}
cout<<"\n";
}
}
void result::sum(){
if((m1!=m2)||(n1!=n2)) {
cout<<"the order should be same for addition";
}
else{
for(i=0;i<m1;i++){
for(j=0;j<n1;j++){
c[i][j]=a[i][j]+b[i][j];
}
}
}
}
void result::sub(){
if((m1!=m2)||(n1!=n2)) {
cout<<"the order should be same for subtraction ";
}
else{
for(i=0;i<m1;i++){
for(j=0;j<n1;j++){
c[i][j]=a[i][j]-b[i][j];
}}}}
void result::mult(void){
if(n2!=m2) {
cout<<"Invalid order limit ";
}
else{
for(i=0;i<m1;i++){
for(j=0;j<n2;j++){
c[i][j]=0;
for(int k=0;k<n1;k++){
c[i][j]+=a[i][k]*b[k][j];
}
} }
}}
int main(){
int ch;
class matrix *p;
class result r;
p=&r;
while(1) {
```

```
cout<<"\n1. Addition of matrices ";
cout<<"\n2. Subtraction of matrices ";
cout<<"\n3. Multipication of matrices ";
cout<<"\n4. Exit";
cout<<"\n Enter your choice: ";
cin>>ch;
switch(ch) {
case 1:p->read();
p->sum();
p->display();
break;
case 2:(p)->read();
p->sub();
p->display();
break;
case 3:p->read();
p->mult();
p->display();
break;
case 4:exit(0);
}}}
```

Output:

```
1. Addition of matrices
2. Subtraction of matrices
3. Multipication of matrices
4. Exit
 Enter your choice: 1

 enter the order of matrix A: 2 2

 enter the elements of matrix A: 1 2 3 4

 enter the order of matrix B: 2 2

 enter the elemnts of matrix B: 5 6 7 8
  6  8
 10 12

1. Addition of matrices
2. Subtraction of matrices
3. Multipication of matrices
4. Exit
 Enter your choice: 4
```

8. Write C++ programs that illustrate how the following forms of inheritance are supported:

a) Single inheritance b) Multiple inheritance c) Multi level inheritance

d) Hierarchical inheritance.

```
#include<iostream>
#include<cmath>
using namespace std;
class top{
public : int a;
void getdata(){
cout<<"Enter the Number : ";
cin>>a;
}
};
class middle :public top{ //single inheritance
public: int b;
void square(){
getdata();
b=a*a;
cout<<"Square of "<<a<<" is :"<<b;
}
};
class bottom :public middle{ //Multi level inheritance
public: int c;
void cube(){
square();
c=b*a;
cout<<"\nCube of "<<a<<" is :"<<c;
}
};
class Squareroot{
public :int num;
void root(int num){
cout<<"\nSquare root of "<<num<<" is : "<<sqrt(num);
}
};
class result: public Squareroot,public bottom{ //Multiple inheritance
public: int x;
void display(){
cube();
x = a;
root(x);
}
};
```

```
int main(){
result b1;
b1.display();
return 0;
}
```

Output:

```
Enter the Number : 5
Square of 5 is :25
Cube of 5 is :125
Square root of 5 is : 2.23607
```

9. a) Write a C++ program that illustrates the order of execution of constructors and destructors when new class derived from more than one base class.

```
#include<iostream>
using namespace std;
class A{
public:A(){
cout<<"\n zero argument constructor of base class a";
}
~A(){
cout<<"\n destructor of base class A";
}
};
class B{
public:B(){
cout<<"\n zero argument constructor of base class b";
}
~B(){
cout<<"\n destructor of base class b";
}
};
class C:public B,A{
public:C(){
cout<<"\n zero argument constructor of desired class c";
}
~C(){
cout<<"\n destructor of class C";
}
};
int main()
{
C obj;
}
```

Output:

```
zero argument constructor of base class b
zero argument constructor of base class a
zero argument constructor of desired class c
destructor of class C
destructor of base class A
destructor of base class b
```

b) Write a Program to Invoking Derived Class Member Through Base Class Pointer.

```
#include <iostream>
using namespace std;
class A{
public: virtual void print_me() {
cout<< "I'm A" <<endl;
}
};
class B : public A{
public: void print_me(){
cout<< "I'm B"<<endl;
}
};
class C : public A{
public:  void print_me(){
cout<< "I'm C" <<endl;
}
};
int main(){
A a;
B b;
C c;
A* p = &a;
p->print_me();
p = &b;
p->print_me();
p = &c;
p->print_me();
return 0;
}
```

Output:

```
I'm Base class A
I'm  Derived class B
I'm  Derived class C
```

10. a) Write a Template Based Program to Sort the Given List of Elements.

```
#include<iostream>
using namespace std;
template<class T>
void bubble(T a[], int n){
int i, j;
for(i=0;i<n-1;i++){
for(j=0;j<n-1;j++){
if(a[j]>a[j+1]) {
T temp;
temp = a[j];
a[j] = a[j+1];
a[j+1] = temp;
}
}
}
}
int main(){
int a[6]={99,58,75,33,29,11};
char b[4]={'z','f','x','a'};
bubble(a,6);
cout<<"\nSorted Order Integers: ";
for(int i=0;i<6;i++)
cout<<a[i]<<"  ";
bubble(b,4);
cout<<"\nSorted Order Characters: ";
for(int j=0;j<4;j++)
cout<<b[j]<<"  ";
}
```

Output:

```
Sorted Order Integers: 11  29  33  58  75  99
Sorted Order Characters: a  f  x  z
```

b) Write a C++ program that uses function templates to find the largest and smallest number in a list of integers and to sort a list of numbers in ascending order.

```
#include<iostream>
using namespace std;
template<class T> //Template declaration
void maxmin(T a[],int n) {//Function Template
int i;
T temp;
for(i=0;i<n;i++)
for(int j=i+1;j<n;j++){
if(a[i]>a[j]) {
temp=a[i];
a[i]=a[j];
a[j]=temp;
}
}
cout<<"max="<<a[n-1]<<"\n"<<"min="<<a[0]<<"\n";
cout<<"sorted list is: ";
for(i=0;i<n;i++)
cout<<a[i]<<" ";
}
int main(){
int a[50],i,ch,n;
double d[50];
float f[50];
char c[50];
cout<<"1.integer"<<endl;
cout<<"2.characters"<<endl;
cout<<"3.float numbers"<<endl;
cout<<"4.double numbers"<<endl;
cout<<"enter corresponding Index Example : enter '1' for integers"<<endl;
cin>>ch;
cout<<"enter the n value: ";
cin>>n;
switch(ch){
case 1:cout<<"enter integers: ";
for(i=0;i<n;i++)
cin>>a[i];
maxmin(a,n);
break;
case 2: cout<<"enter characters: ";
for(i=0;i<n;i++)
cin>>c[i];
maxmin(c,n);
```

```
break;
case 3: cout<<"enter floatnumbers: ";
for(i=0;i<n;i++)
cin>>f[i];
maxmin(f,n);
break;
case 4: cout<<"enter doublenumbers: ";
for(i=0;i<n;i++)
cin>>d[i];
maxmin(d,n);
break;
default:cout<<"Invalid choice entered...";
}
return 0;
}
```

Output:

```
1.integer
2.characters
3.float numbers
4.double numbers
enter corresponding Index Example : enter '1' for integers
1
enter the n value: 5
enter integers: 5 4 6 2 1
max=6
min=1
sorted list is: 1 2 4 5 6
```

11. a) Write a Program Containing a Possible Exception. Use a Try Block to Throw it and a Catch Block to Handle it Properly.

```
#include <iostream>
using namespace std;
int main(){
int x = -1;
cout << "Before try \n";
try {
cout << "Inside try \n";
if (x < 0){
throw x;
cout << "After throw (Never executed) \n";
}
}
catch (int x ) {
cout << "Exception Caught \n";
}
cout << "After catch (Will be executed) \n";
return 0;
}
```

Output:

```
Before try
Inside try
Exception Caught
After catch (Will be executed)
```

b) Write a Program to Demonstrate the Catching of All Exceptions.

```
#include <iostream>
using namespace std;
int main(){
try  {
throw 10;
}
catch (char excp){
cout << "Caught " << excp;
}
catch (...){
cout << "Default Exception\n";
}
return 0;
}
```

Output:

```
Default Exception
```