**Name: Om Ramakant Parvate Roll no.: 168 Div.: B Batch:B6**

**Class: FYBCA**

**Practical-13:**

#include<iostream>

using namespace std;

class A {

public:

virtual void display() {

cout << "Base class is invoked" << endl;

}

};

class B : public A {

public:

void display() override {

cout << "Derived Class is invoked" << endl;

}

};

int main() {

A \*a;

B b;

a = &b;

a->display();

return 0;

}

**Output:-**

Derived Class is invoked

--------------------------------

**Name: Om Ramakant Parvate Roll no.: 168 Div.: B Batch:B6**

**Class: FYBCA**

**Practical-14 (A): Write a program to demonstrate the use of a pointer to pointer.**

#include<iostream>

using namespace std;

int main() { int age = 30;

int \*ptr1 = &age;

int \*\*ptr2 = &ptr1;

cout<<"Value stored in the age variable "<<age<<"\n";

cout<<"Value accessed using single pointer "<<\*ptr1<<"\n";

cout<<"Value accessed using double pointer "<<\*\*ptr2<<"\n";

return 0; }

**Output:**

Value stored in the age variable 30

Value accessed using single pointer 30

Value accessed using double pointer 30

**Name: Om Ramakant Parvate Roll no.: 168 Div.: B Batch:B6**

**Class: FYBCA**

**Practical-14 (B): Write a program to create pointers that point to objects and access their members.**

#include <iostream>

using namespace std;

class Vehicle {

public:

string type;

string maker;

int year;

void show() {

cout << "The vehicle type is a " << type

<< ", built by " << maker<< " and Year is " << year << endl;

}

};

int main() {

Vehicle veh;

Vehicle \*vehPtr = &veh;

vehPtr->type = "CAR";

vehPtr->maker = "RR";

vehPtr->year = 2025;

vehPtr->show();

return 0;

}

**Output:**

The vehicle type is a CAR, built by RR and Year is 2025

**Name: Om Ramakant Parvate Roll no.: 168 Div.: B Batch:B6**

**Class: FYBCA**

**Practical-14 (C): C++ program to implementation**

#include <iostream>

using namespace std;

int multiply(int a, int b) { return a \* b; }

int main() {

int (\*func)(int, int);

func = multiply;

int prod = func(60, 3);

cout << "The value of the product is: " << prod << endl;

return 0;

}

**Output:**

The value of the product is: 180

**Name: Om Ramakant Parvate Roll no.: 168 Div.: B Batch:B6**

**Class: FYBCA**

**Practical-15: Write a program to demonstrate use of Exception Handling.**

int main() {

double numerator, denominator, divide;

cout << "Enter numerator: ";

cin >> numerator;

cout << "Enter denominator: ";

cin >> denominator;

try {

if (denominator == 0)

throw 0;

divide = numerator / denominator;

cout << numerator << " / " << denominator << " = " << divide << endl;

}

catch (int num\_exception) {

cout << "Error: Cannot divide by " << num\_exception << endl;

}

return 0;

}

**Output:**

Enter numerator: 5

Enter denominator: 6

5 / 6 = 0.833333

**Name: Om Ramakant Parvate Roll no.: 168 Div.: B Batch:B6**

**Class: FYBCA**

**Practical-16(A): Write a program to find the largest and smallest elements from an array**

#include<iostream>

using namespace std;

int main ()

{

int arr[10], n, i, max, min;

cout << "Enter the size of the array : ";

cin >> n;

cout << "Enter the elements of the array : ";

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

cin >> arr[i];

max = arr[0];

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

{

if (max < arr[i])

max = arr[i];

}

min = arr[0];

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

{

if (min > arr[i])

min = arr[i];

}

cout << "Largest element : " << max;

cout << "Smallest element : " << min;

return 0;

}

**Output:**

Enter the size of the array : 3

Enter the elements of the array : 6

7

8

Largest element : 8Smallest element : 6

**Name: Om Ramakant Parvate Roll no.: 168 Div.: B Batch:B6**

**Class: FYBCA**

**Practical-16(B): Write a program to sort an array in ascending and descending order**.

#include<iostream>

using namespace std;

void ascDecFunc(int a[], int n)

{

int temp;

for(int i=0;i < n-1;i++)

{

for(int j = 0;j < n/2; j++) { if(a[j]>a[j+1])

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

for(int j = n/2;j < n-1; j++)

{

if(a[j] < a[j+1])

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

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

cout<<a[i]<<" ";

}

int main()

{

int arr[] = {3, 2, 4, 1, 10, 30, 40, 20};

int len = sizeof(arr) / sizeof(arr[0]);

ascDecFunc(arr, len);

return 0;

}

**Output:**

1 2 3 4 40 30 20 10

**Name: Om Ramakant Parvate Roll no.: 168 Div.: B Batch:B6**

**Class: FYBCA**

**Practical-17: Write a program to concatenate two strings and find the length of a string**

#include <iostream>

#include <cstring>

using namespace std;

int

cout << "Enter the first string: ";

cin.getline(str1, 100);

cout << "Enter the second string: ";

cin.getline(str2, 100);

main() {

char str1[100], str2[100], result[200];

strcpy(result, str1);

strcat(result, str2);

cout << "Concatenated String: " << result << endl;

cout << "Length of Concatenated String: " << strlen(result) << endl;

return 0;

}

**Output:**

Enter the first string: 50

Enter the second string: 60

Concatenated String: 5060

Length of Concatenated String: 4

**Name: Om Ramakant Parvate Roll no.: 168 Div.: B Batch:B6**

**Class: FYBCA**

**Practical-18: Write a program to calculate the factorial of a number using recursion**

#include<iostream>

using namespace std;

int factorial(int n);

int main() {

int n;

cout << "Enter a positive integer: ";

cin >> n;

cout << "Factorial of " << n << " = " << factorial(n);

return 0;

}

int factorial(int n) {

if(n > 1)

return n \* factorial(n - 1);

else

return 1;

}

**Output:**

Enter a positive integer: 6

Factorial of 6 = 720

**Name: Om Ramakant Parvate Roll no.: 168 Div.: B Batch:B6**

**Class: FYBCA**

**Practical-19: Write a program that demonstrates different types of polymorphism (e.g.,method overriding).**

#include <iostream>

using namespace std;

class Animal {

public:

void eat(){

cout<<"Eating...";

}

};

class Dog: public Animal

{

public:

void eat()

{

cout<<"Eating bread...";

}

};

int main(void) {

Dog d = Dog();

d.eat();

return 0;

}

**Output:**

Eating bread...

**Name: Om Ramakant Parvate Roll no.: 168 Div.: B Batch:B6**

**Class: FYBCA**

**Practical-21: Write a program to allocate and deallocate memory dynamically using pointers.**

#include <iostream>

using namespace std;

int main()

{

int\* arr = new int[5];

cout << "Enter 5 values in the array: ";

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

{

cin >> arr[i];

}

cout << "\nArray elements: ";

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

{

cout << arr[i] << " ";

}

delete [] arr;

cout << "\nGarbage array values after deallocation of array memory: ";

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

{

cout << arr[i] << " ";

}

return 0;

}

**Output:**

Enter 5 values in the array: 5

6

7

9

7

Array elements: 5 6 7 9 7

Garbage array values after deallocation of array memory: 11950688 0 11927888 0 7