**/\* THIS C++ PROGRAM ILLUSTRATES THE CONCEPT OF**

**\* STATIC DATA MEMBERS (STATIC DATA CLASS) \*/**

**/\*NAME : SAGAR GIRI, SECTION: A, ROLL NO. 205 \*/**

#include <iostream>

using namespace std;

class Distance

{

private:

int feet; float inches;

public:

static int count; **//static data member**

Distance()

{

feet = 0;inches = 0.0;

count++; **//increments count for every object created**

}

Distance (int ft, float in)

{

feet = ft;inches = in;

count++; **//increments count for every object created**

}

void display()

{

cout<<feet<<"\'-"<<inches<<"\"";

}

}; **//end class Distance**

int Distance::count = 0; **//definition of static variable count**

int main()

{

Distance d1(5,7.6),d2(4,3.9),d3;

cout<<endl<<"Total Objects="<<Distance::count;

cout<<endl<<"Distance One =";d1.display();

cout<<endl<<"Distance Two =";d2.display();

cout<<endl<<"Distnce Three=";d3.display();

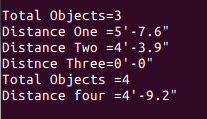
Distance d4(4,9.2);

cout<<endl<<"Total Objects ="<<Distance::count;

cout<<endl<<"Distance four =";d4.display();

}

OUTPUT:



**/\* THIS PROGRAM ILLUSTRATES THE CONCEPT OF "CONST" QUALIFIER**

**\* "CONST" IS A KEYWORD IN C++ \*/**

**/\*NAME : SAGAR GIRI, SECTION: A, ROLL NO. 205 \*/**

#include <iostream>

using namespace std;

class Distance

{

private:

int feet; float inches;

public:

Distance()

{

feet = 0;inches=0.00;

}

Distance (int ft, float in)

{

feet = ft; inches = in;

}

void display() const **//constant display member function**

{

cout<<feet<<"\'-"<<inches<<"\""<<endl;

**//here we cannot do feet++ or inches++ but can change the**

}

}; **//end class Distance**

int main()

{

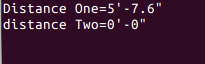
Distance d1(5,7.6),d2;

cout<<"Distance One=";d1.display();

cout<<"distance Two=";d2.display();

}

OUTPUT:



**/\* THIS PROGRAM ILLUSTRATES THE CONCEPT OF**

**\* PASSING ARGUMENTS BY REFERENCE IN A MEMBER FUNCTION \*/**

**/\* NAME : SAGAR GIRI, ROLL : 205, SECTION : A\*/**

#include <iostream>

using namespace std;

void exchange(int&, int&);

int main()

{

int a = 3, b = 4;

cout << "before exchange";

cout << "a = " << a << endl <<"b = " << b;

exchange(a, b);

cout << endl << "after exchange";

cout << "a = " << a << endl << "b = " << b;

}

void exchange(int& x, int& y)

{

int t;

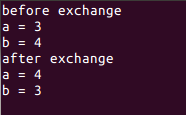
t = x;

x = y;

y = t;

}

OUTPUT:



**/\* THIS PROGRAM ILLUSTRATES THE CONCEPT OF**

**\* INLINE MEMBER FUNCTION \*/**

**/\* NAME : SAGAR GIRI, ROLL : 205, SECTION : A\*/**

#include <iostream>

using namespace std;

class Distance

{

private:

int feet;float inches;

public:

Distance()

{

feet = 0;

inches = 0.0;

}

Distance(int ft, float in)

{

feet = ft;

inches = in;

}

inline Distance addDistance(Distance dd1) **//defining inline function**

{

Distance temp;

temp.feet = feet + dd1.feet;

temp.inches = inches + dd1.inches;

if(inches >= 12.0)

{

inches -= 12.0;

feet++;

}

return temp;

}

void display()

{

cout << feet << "\'-" << inches << "\"" << endl;

}

};

int main()

{

Distance d1(5, 6.7), d2(7, 3.2), d3;

d3 = d1.addDistance(d2);

d3.display();

}

OUTPUT:



**/\* THIS PROGRAM ILLUSTRATES THE CONCEPT OF**

**\* PASSING AS POINTER IN A MEMBER FUNCTION \*/**

**/\* NAME : SAGAR GIRI, ROLL : 205, SECTION : A\*/**

#include <iostream>

using namespace std;

class exchange

{

private:

int a;

int b;

public:

exchange(int x, int y) **//two argument constructors**

{

a = x;

b = y;

}

void exch(exchange\* c1) **//swap the value of a and b using pointer**

{

int temp=0;

temp = c1->a;

c1->a = c1->b;

c1->b = temp;

}

void display1()

{

cout<<"before exchange"<<endl;

cout<<"a = "<<a<<endl<<"b = "<<b<<endl;

}

void display2()

{

cout<<"after exchange"<<endl;

cout<<"a = "<<a<<endl<<"b = "<<b;

}

}; **//end class exchange**

int main()

{

exchange c1(3,4);

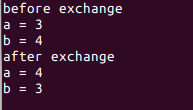
c1.display1();

c1.exch(&c1); **//passing address of the object in member function**

c1.display2();

return 0;

}

OUTPUT:

**/\* THIS PROGRAM ILLUSTRATES THE CONCEPT OF**

**\* PASSING DEFAULT ARGUMENTS IN A MEMBER FUNCTION \*/**

**/\* NAME : SAGAR GIRI, ROLL : 205, SECTION : A\*/**

#include <iostream>

using namespace std;

void repchar(char = '#', int = 30); **//Function Prototype**

int main()

{

repchar();

repchar('$');

repchar('\*', 20);

return 0;

}

void repchar(char ch, int n)

{

cout << endl;

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

{

cout << ch;

}

}

OUTPUT:

