\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ASSIGNMENT 8\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NAME: RAMAVATH SANTHOSH ROLL NO: 22MCF1R40 DEPARTMENT: MCA (CSE) COURSE: PSP

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\* 1. Write a program to swap the values two integer members of different classes

using friend function.

\*/

#include <iostream>

using namespace std;

class B;

class A

{

int data;

public:

A(int n)

{

data = n;

}

friend void swap(A&, B&);

void show\_data()

{

cout << "A = " << data << endl;

}

};

class B

{

int data;

public:

B(int n){ data = n; }

friend void swap(A&, B&);

void show\_data(){ cout << "B = " << data << endl; }

};

void swap(A &a, B &b)

{

int temp = a.data;

a.data = b.data;

b.data = temp;

}

int main()

{

A a(10);

B b(20);

cout << "\*\*\*\*\*\*\*\*\*Before swapping..." << endl;

a.show\_data();

b.show\_data();

swap(a, b);

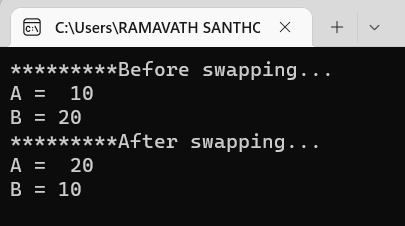
cout << "\*\*\*\*\*\*\*\*\*After swapping..." << endl;

a.show\_data();

b.show\_data();

return 0;

}



/\*

2. Write a program for addition of two complex numbers using friend function (use

constructor function to initialize data members of complex class).

\*/

#include <iostream>

using namespace std;

class complex

{

int real, img;

public:

complex(int a, int b)

{

real = a;

img = b;

}

friend complex add(complex, complex);

void show()

{

cout << real << " + " << img << "i" << endl;

}

};

complex add(complex c1, complex c2)

{

return complex(c1.real + c2.real, c1.img + c2.img);

}

int main()

{

complex c1(5, 2), c2(11, 8);

c1.show();

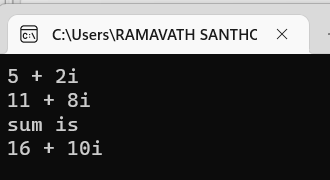
c2.show();

cout << "sum is" << endl;

add(c1, c2).show();

return 0;

}



/\*

3. Define a class string and overload == to compare two strings and + operator for

concatenation of two strings.

\*/

#include <iostream>

using namespace std;

class myString

{

string s;

public:

myString(string str)

{

s = str;

}

void show()

{

cout << s << endl;

}

bool operator == (myString s1)

{

return s == s1.s;

}

myString operator + (myString s1)

{

return myString(s + s1.s);

}

};

int main()

{

myString s1("ayush");

myString s2("Ayush");

s1.show();

s2.show();

cout << "Comparing the two strings\n";

cout << (s1 == s2 ? "strings are equal": "strings are not equal") << endl;

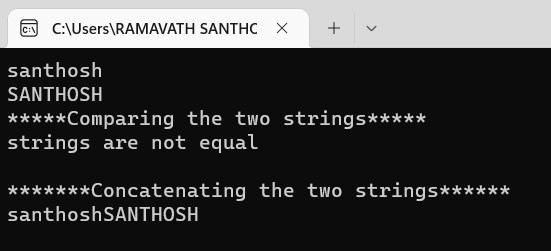
cout << "Concatenating the two strings\n";

myString s3 = s1 + s2;

s3.show();

return 0;

}



/\*

4. Write a program to perform matrix addition using operator overloading concept.

Matrix

a[100][100], m,n

void getdata()

void show()

matrix operator+(matrix &amp;x,matrix &amp;y)

\*/

#include <iostream>

using namespace std;

class matrix

{

int a[100][100], m, n;

public:

void getdata()

{

cout << "Enter the number of rows: "; cin >> m;

cout << "Enter the number of columns: "; cin >> n;

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

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

cin >> a[i][j];

}

void show()

{

for(int i = 0; i < m; i++){

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

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

cout << endl;

}

}

matrix operator + (matrix m1)

{

matrix m3;

m3.m = m;

m3.n = n;

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

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

m3.a[i][j] = a[i][j] + m1.a[i][j];

return m3;

}

};

int main()

{

matrix m1, m2;

m1.getdata();

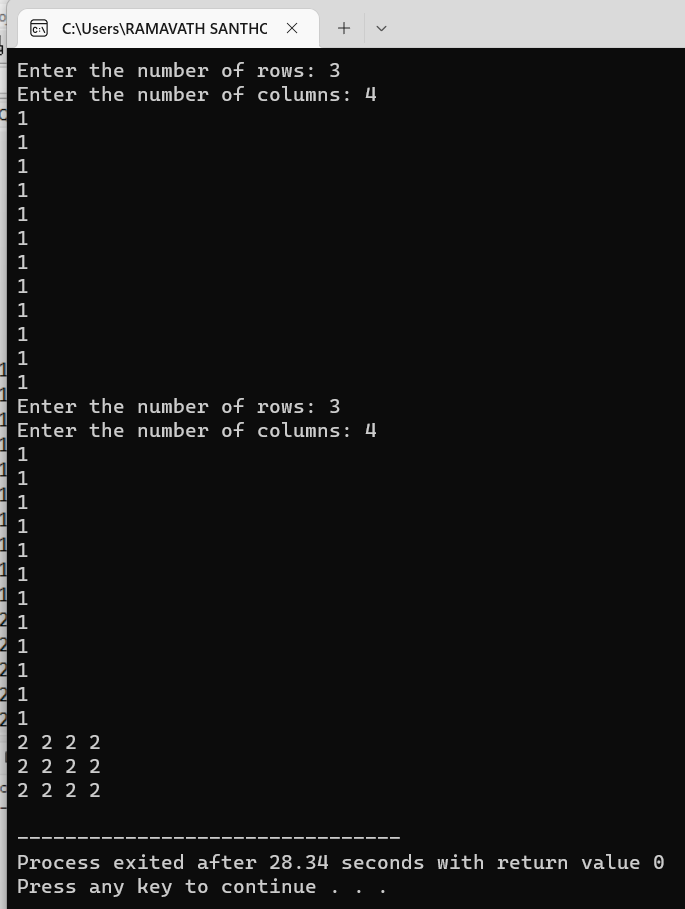
m2.getdata();

matrix m3 = m1 + m2;

m3.show();

return 0;

}



/\*

5. Write a program to maintain the records of person with details (name and age) and

find the eldest among them. The program must use this pointer to return the result.

\*/

#include <iostream>

using namespace std;

class Person

{

private:

string name;

int age;

public:

Person();

int get\_age()

{

return age;

}

void set\_data(string s, int n)

{

name = s;

age = n;

}

void show\_data()

{

cout << "Name: " << name << endl;

cout << "Age: " << age << endl;

}

};

Person::Person()

{

name = "";

age = -1;

}

class Record

{

private:

Person \*arr;

int n;

public:

Record(int sz);

void show\_eldest();

};

Record::Record(int sz)

{

n = sz;

arr = new Person[n];

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

{

string name;

int age;

cout << "Enter name of Person " << i + 1 << ": ";

cin >> name;

cout << "Enter age of Person " << i + 1 << ": ";

cin >> age;

arr[i].set\_data(name, age);

}

}

void Record::show\_eldest()

{

int max\_idx = 0;

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

if(this->arr[i].get\_age() > this->arr[max\_idx].get\_age())

max\_idx = i;

cout << "Eldest person is " << endl;

this->arr[max\_idx].show\_data();

}

int main()

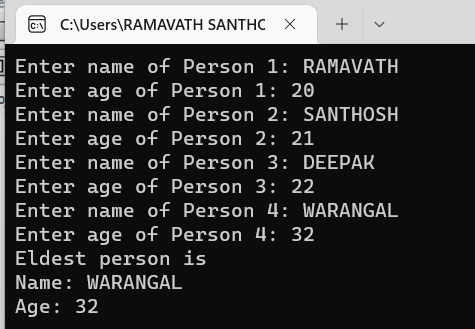
{

Record record(4);

record.show\_eldest();

return 0;

}



/\*

6. Write a C++ program to count the number of persons inside a bank, by increasing count

whenever a person enters a bank, using an increment(++) operator overloading function, and

decrease the count whenever a person leaves the bank using a decrement(--) operator

overloading function inside a class

\*/

#include <iostream>

using namespace std;

class Counter

{

private:

int count;

public:

Counter();

Counter operator++(int)

{

Counter c = \*this;

count++;

return c;

}

Counter operator--(int)

{

Counter c = \*this;

count--;

return c;

}

void show() { cout << "Number of persons: " << count << endl; }

};

Counter::Counter()

{

int count = 0;

}

int main()

{

Counter c;

while (1)

{

cout << "1. Person Entering Bank" << endl;

cout << "2. Person Leaving Bank" << endl;

cout << "3. Show Number of People in Bank" << endl;

cout << "0. Exit Program" << endl;

cout << "\nEnter choice" << endl;

int choice;

cin >> choice;

if (!choice)

break;

switch (choice)

{

case 1:

c++;

break;

case 2:

c--;

break;

case 3:

c.show();

break;

default:

cout << "Wrong choice..." << endl;

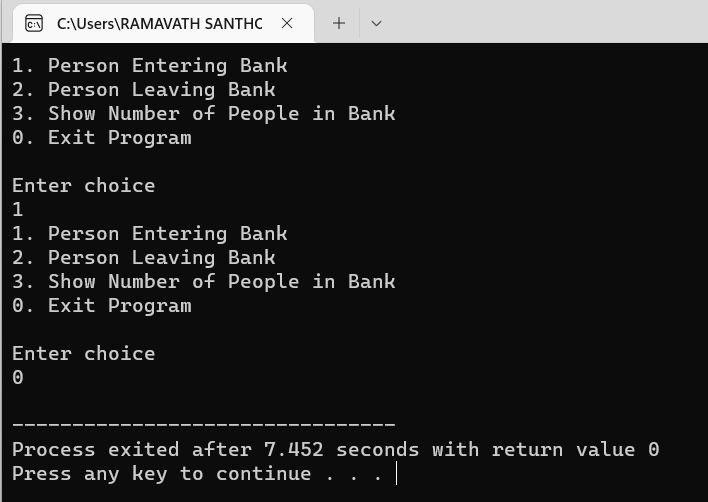
break;

}

}

return 0;

}



/\*

7. Write a program to accept the student detail such as name and 3 different marks by get\_data()

method and display the name and average of marks using display() method. Define a friend

class for calculating the average of marks using the method marrk\_avg().

\*/

#include <iostream>

using namespace std;

class Average;

class Student

{

private:

string name;

int marks[3];

public:

Student()

{

name = "";

marks[0] = marks[1] = marks[2] = 0;

}

void get\_data()

{

cout << "Enter the name of student: ";

cin >> name;

cout << "Enter the marks in subject 1: ";

cin >> marks[0];

cout << "Enter the marks in subject 2: ";

cin >> marks[1];

cout << "Enter the marks in subject 3: ";

cin >> marks[2];

}

friend class Average;

};

class Average

{

public:

float show\_average(Student s)

{

return (s.marks[0] + s.marks[1] + s.marks[2]) / 3.0;

}

};

int main()

{

Student s;

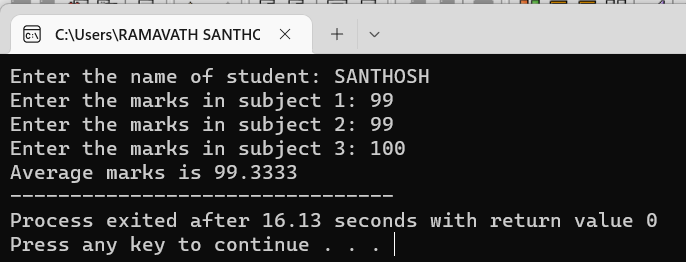
s.get\_data();

Average a;

cout << "Average marks is " << a.show\_average(s);

return 0;

}



\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*