**Question#1**

#include<iostream>

#include<string>

using namespace std;

template<typename t>

class studentdata {

private:

t name;

t rollno;

t section;

string\* list;

int size;

public:

studentdata();

~studentdata();

void setname(t name);

void setsection(t section);

void setrollno(int rollNo);

void setlist(t\* list, int listsize);

t getname();

t getsection();

int getrollno();

t\* getlist();

int getsize();

};

template<typename t>

studentdata<t>::studentdata()

{

}

template<typename t>

studentdata<t>::~studentdata()

{

if (list != nullptr)

{

delete[]list;

list = nullptr;

}

}

template<typename t>

void studentdata<t>::setname(t name)

{

this->name = name;

}

template<typename t>

void studentdata<t>::setsection(t section)

{

this->section = section;

}

template<typename t>

void studentdata<t>::setrollno(int rollno)

{

this->rollno = rollno;

}

template<typename t>

void studentdata<t>::setlist(t\* list, int listsize)

{

if (this->list != nullptr)

{

delete[] this->list;

this->list = nullptr;

}

this->list = new t[listsize];

size = listsize;

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

{

this->list[i] = list[i];

}

}

template<typename t>

t studentdata<t>::getname()

{

return name;

}

template<typename t>

int studentdata<t>::getrollno()

{

return rollno;

}

template<typename t>

t studentdata<t>::getsection()

{

return section;

}

template<typename t>

int studentdata<t>::getsize()

{

return size;

}

template<typename t>

t\* studentdata<t>::getlist()

{

return list;

}

//prime number

template<typename T>

T checker(T number, T end)

{

int count = 0;

for (int j = 1; j < end; ++j)

{

if (number % j == 0)

{

++count;

}

}

return count;

}

template<typename T>

void Primenumber(T start, T end)

{

cout << "Prime Numbers are : " << endl;

for (int i = start; i < end; ++i)

{

int count = checker(i, end);

if (count < 3)

{

cout << i << " ";

}

}

}

//base ball

template<typename t1, typename t2>

class baseball

{

private:

t1 name;

t2 homeruns;

public:

baseball()

{

}

void setname(t1 name)

{

this->name = name;

}

void sethomeruns(t2 homeruns)

{

this->homeruns = homeruns;

}

t1 getname()

{

return name;

}

t2 gethomeruns()

{

return homeruns;

}

};

// calculator

template<typename t>

class calculation

{

public:

calculation()

{}

t addition(t num1, t num2);

t subtraction(t num1, t num2);

t multiplication(t num1, t num2);

};

template<typename t>

t calculation<t>::addition(t num1, t num2)

{

return num1 + num2;

}

template<typename t>

t calculation<t>::subtraction(t num1, t num2)

{

return num1 - num2;

}

template<typename t>

t calculation<t>::multiplication(t num1, t num2)

{

return num1 \* num2;

}

int main()

{

studentdata<string> s;

string name, section;

string\* list;

int rollno;

// data of student

cout << "Enter name of student = ";

getline(cin, name);

cout << "Enter section name = ";

getline(cin, section);

cout << "Enter rollno = ";

cin >> rollno;

cin.ignore();

int num;

cout << "Enter number of courses = ";

cin >> num;

list = new string[num];

cin.ignore();

cout << "enter number of subjects : " << endl;

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

{

cout << i + 1 << " Subject = ";

getline(cin, list[i]);

}

s.setname(name);

s.setsection(section);

s.setrollno(rollno);

s.setlist(list, num);

cout << endl;

// data of baseball

int n;

cout << "enter no of player of baseball = ";

cin >> n;

baseball<string, int>\* b;

b = new baseball<string, int>[n];

int count = 0;

bool exit = false;

while (!exit)

{

int choice;

cout << "Enter 1 for student data" << endl;

cout << "Enter 2 to diplay prime number" << endl;

cout << "Enter 3 for base ball information" << endl;

cout << "Enter 4 for calculation " << endl;

cout << "Enter 5 to exit program " << endl;

cout << endl;

cout << "Enter choice = ";

cin >> choice;

cout << endl;

if (choice == 1)

{

cout << endl;

cout << "Name = " << name << endl;

cout << "section = " << section << endl;

cout << "rollno = " << rollno << endl;

cout << "subject = " << endl;;

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

{

cout << i + 1 << " Subject = " << list[i];

cout << endl;

}

cout << endl;

}

else if (choice == 2)

{

int firstno;

int secondno;

cout << "Enter first no = ";

cin >> firstno;

cout << "Enter second no = ";

cin >> secondno;

Primenumber(firstno, secondno);

cout << endl;

}

else if (choice == 3)

{

int select;

cout << "Enter 1 to add player" << endl;

cout << "Enter 2 to update player" << endl;

cout << "Enter 3 to display" << endl;

cout << "enter choice = ";

cin >> select;

if (select == 1)

{

if (count >= n)

{

cout << "Full! can't add player" << endl;;

}

string name1;

int homeruns;

cin.ignore();

cout << "Enter name = ";

getline(cin, name1);

cout << "enter no of homeruns = ";

cin >> homeruns;

b[count].sethomeruns(homeruns);

b[count].setname(name);

count++;

}

else if (select == 2)

{

int n1;

cout << "Enter number of player you want to update" << endl;

cin >> n1;

if (n1 > count)

{

cout << "Out of range" << endl;

}

string name1;

int homeruns1;

cin.ignore();

cout << "Enter name = ";

getline(cin, name1);

cout << "enter no of homeruns = ";

cin >> homeruns1;

b[n1].setname(name1);

b[n1].sethomeruns(homeruns1);

}

else if (select == 3)

{

if (count == 0)

{

cout << "NO players information" << endl;

}

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

{

cout << "Information of player 01 : " << endl;;

cout << "name = " << b[i].getname() << endl;

cout << "Home runs = " << b[i].gethomeruns() << endl;

}

}

}

else if (choice == 4)

{

calculation<int> c;

int w = 0;

cout << endl;

cout << "Enter 1 for addition" << endl;

cout << "Enter 2 for subtraction" << endl;

cout << "Enter 3 for multiplication" << endl;

cout << endl;

cout << "Enter choice = ";

cin >> w;

cout << endl;

if (w == 1)

{

cout << "Addition = " << c.addition(10, 20) << endl;

}

else if (w == 2)

{

cout << "SUbtraction = " << c.subtraction(20, 10) << endl;

}

else if (w == 3)

{

cout << "Multiplication = " << c.multiplication(10, 7) << endl;

}

else

{

cout << "invalid number " << endl;

}

cout << endl;

}

else if (choice == 5)

{

exit = true;

}

else

{

cout << "Invalid output" << endl;

}

}

delete[]list;

list = nullptr;

delete[]b;

b = nullptr;

system("pause");

return 0;

}

****

**Question#2**

#include<iostream>

#include<string>

using namespace std;

class invaliddate

{

string massage;

public:

invaliddate()

{

massage = "Invalid date entered\n";

}

string what()

{

return massage;

}

};

class invalidmonth

{

string output;

public:

invalidmonth()

{

output = "Invalid month entered\n";

}

string what()

{

return output;

}

};

int main()

{

int month;

int days;

int year;

try

{

cout << "Enter days = ";

cin >> days;

cout << "Enter month = ";

cin >> month;

cout << "Year = ";

cin >> year;

if (month > 12 && month < 1)

{

throw invalidmonth();

}

}

catch (invalidmonth ex)

{

cout << ex.what() << endl;

}

try

{

if (month == 2)

{ // February

if ((year % 400 == 0) || (year % 4 == 0 && year % 100 != 0))

{ // Leap year

if (days > 29)

{

throw invaliddate();

}

}

else

{

if (days > 28)

{

throw invaliddate();

}

}

}

if (days < 1 || days > 31)

{

throw invaliddate();

}

else if ((month == 4 || month == 6 || month == 9 || month == 11) && days > 30)

{

throw invaliddate();

}

string monthname;

if (month == 1)

{

monthname = "January";

}

else if (month == 2)

{

monthname = "February";

}

else if (month == 3)

{

monthname = "March";

}

else if (month == 4)

{

monthname = "April";

}

else if (month == 5)

{

monthname = "May";

}

else if (month == 6)

{

monthname = "June";

}

else if (month == 7)

{

monthname = "july";

}

else if (month == 8)

{

monthname = "August";

}

else if (month == 9)

{

monthname = "September";

}

else if (month == 10)

{

monthname = "October";

}

else if (month == 11)

{

monthname = "November";

}

else if (month == 4)

{

monthname = "December";

}

cout << "DATE = " << monthname << " - " << days << " - " << year << endl;

}

catch (invaliddate d)

{

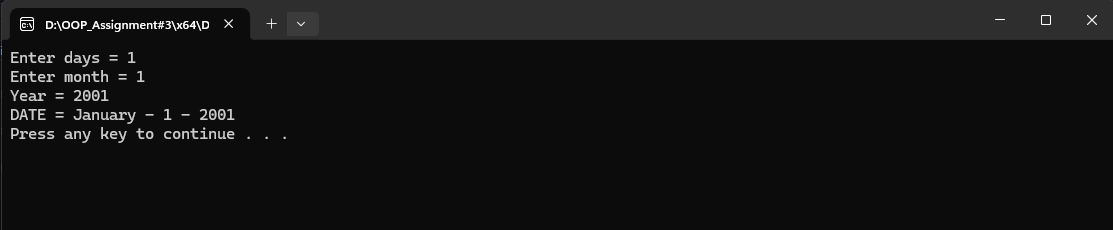
cout << d.what() << endl;

}

system("pause");

return 0;

}

****

**Question#3**

#include<iostream>

#include<string>

using namespace std;

void function\_menu(int& choice)

{

cout << "1. for addition" << endl;

cout << "2. for subtarction" << endl;

cout << "3. for multiplication" << endl;

cout << "4. for division" << endl;

cout << "5. for exit program" << endl;

cout << "Your choice = ";

cin >> choice;

}

void add\_fractions() {

int num1, dem1, num2, dem2;

cout << "Enter numerator 1: ";

cin >> num1;

cout << "Enter denominator 1: ";

cin >> dem1;

cout << "Enter numerator 2: ";

cin >> num2;

cout << "Enter denominator 2: ";

cin >> dem2;

if (dem1 == 0 || dem2 == 0)

throw 0;

int denominator = dem1 \* dem2;

int numerator = (dem2 \* num1) + (dem1 \* num2);

cout << "Result after addition: " << numerator << "/" << denominator << endl;

}

void sub\_fractions()

{

int num1, dem1, num2, dem2;

cout << "Enter numerator 1: ";

cin >> num1;

cout << "Enter denominator 1: ";

cin >> dem1;

cout << "Enter numerator 2: ";

cin >> num2;

cout << "Enter denominator 2: ";

cin >> dem2;

if (dem1 == 0 || dem2 == 0)

throw 0;

int denominator = dem1 \* dem2;

int numerator = (dem2 \* num1) - (dem1 \* num2);

cout << "Result after subtraction: " << numerator << "/" << denominator << endl;

}

void mul\_fractions() {

int num1, dem1, num2, dem2;

cout << "Enter numerator 1: ";

cin >> num1;

cout << "Enter denominator 1: ";

cin >> dem1;

cout << "Enter numerator 2: ";

cin >> num2;

cout << "Enter denominator 2: ";

cin >> dem2;

if (dem1 == 0 || dem2 == 0)

throw 0;

int denominator = dem1 \* dem2;

int numerator = num1 \* num2;

cout << "Result after multiplication: " << numerator << "/" << denominator <<

endl;

}

void div\_fractions()

{

int num1, dem1, num2, dem2;

cout << "Enter numerator 1: ";

cin >> num1;

cout << "Enter denominator 1: ";

cin >> dem1;

cout << "Enter numerator 2: ";

cin >> num2;

cout << "Enter denominator 2: ";

cin >> dem2;

if (dem1 == 0 || dem2 == 0)

throw 0;

int denominator = dem1 \* num2;

int numerator = dem2 \* num1;

cout << "Result after division: " << numerator << "/" << denominator << endl;

}

int main()

{

int choice, num1, num2;

bool exit = false;

while (!exit) {

function\_menu(choice);

try {

switch (choice) {

case 1:

{

add\_fractions();

break; }

case 2:

{

sub\_fractions();

break;

}

case 3:

{

mul\_fractions();

break;

}

case 4:

{

div\_fractions();

break;

}

case 5:

{

exit = true;

}

}

}

catch (int a) {

cout << "denominator is zero" << endl;

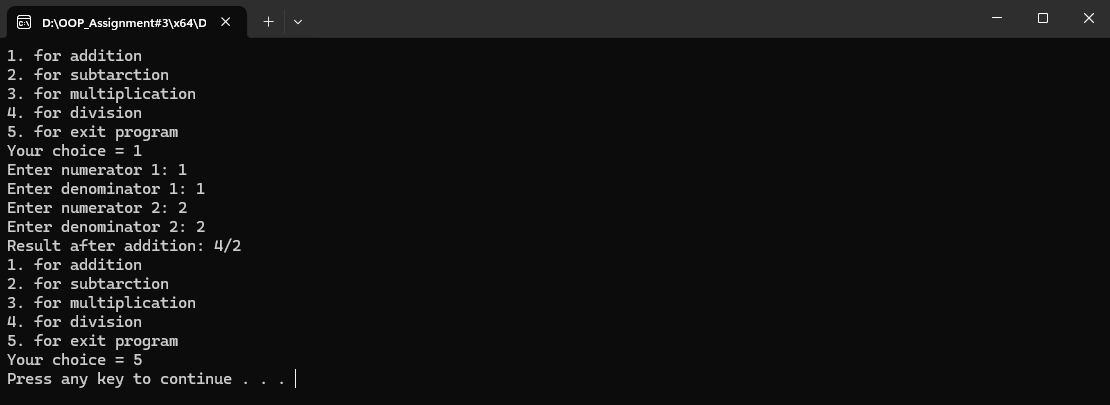
}

}

system("pause");

return 0;

}

****

**Question#4**

#include <iostream>

#include <string>

using namespace std;

const int MAX\_EXITS = 4;

const int MAX\_ITEMS = 10;

class Location

{

private:

string description;

struct Exit {

string direction;

Location\* location;

};

Exit exits[MAX\_EXITS];

string items[MAX\_ITEMS];

int numExits;

int numItems;

public:

Location(const string& desc) : description(desc), numExits(0), numItems(0) {}

void setExit(const string& direction, Location\* location) {

exits[numExits++] = { direction, location };

}

void addItem(const string& item) {

items[numItems++] = item;

}

string getDescription() const { return description; }

Location\* getNextLocation(const string& direction) const {

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

if (exits[i].direction == direction) {

return exits[i].location;

}

}

return nullptr;

}

bool hasItem(const string& item) const {

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

if (items[i] == item) {

return true;

}

}

return false;

}

bool removeItem(const string& item) {

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

if (items[i] == item) {

for (int j = i; j < numItems - 1; ++j) {

items[j] = items[j + 1];

}

numItems--;

return true;

}

}

return false;

}

};

class Player

{

private:

Location\* location;

string inventory[MAX\_ITEMS];

int numItems;

public:

Player(Location\* startLoc) : location(startLoc), numItems(0) {}

void look() const {

cout << location->getDescription() << endl;

}

void showInventory() const {

if (numItems == 0) {

cout << "You are not carrying anything." << endl;

}

else {

cout << "Inventory: ";

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

cout << inventory[i] << " ";

}

cout << endl;

}

}

void move(const string& direction) {

Location\* nextLoc = location->getNextLocation(direction);

if (nextLoc) {

location = nextLoc;

cout << "You moved to " << location->getDescription() << "." << endl;

}

else {

cout << "You can't go that way." << endl;

}

}

};

int main() {

Location hallway("A dark and dusty hallway.");

Location kitchen("A well-lit kitchen with a giant oven.");

Location bedroom("A cozy bedroom with a fluffy bed.");

// Set exits

hallway.setExit("north", &kitchen);

kitchen.setExit("south", &hallway);

kitchen.setExit("east", &bedroom);

bedroom.setExit("west", &kitchen);

// Add items

kitchen.addItem("knife");

bedroom.addItem("pillow");

// Create player

Player player(&hallway);

bool playing = false;

while (!playing) {

cout << "> ";

string command;

getline(cin, command);

if (command == "look") {

player.look();

}

else if (command == "inventory") {

player.showInventory();

}

else if (command == "north" || command == "south" ||

command == "east" || command == "west") {

player.move(command);

}

else if (command == "exit")

{

playing = true;

}

else {

cout << "I don't understand that command." << endl;

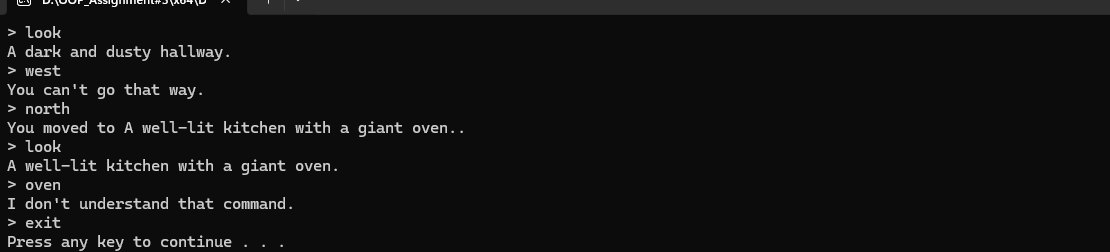
}

}

system("pause");

return 0;

}

****

**Question#5**

#include<iostream>

#include<fstream>

#include<string>

using namespace std;

template<typename T>

class Course;

template<typename T>

class Student

{

private:

T id;

string full\_name;

T unique\_identity;

Course<T>\* ptr[10];

int course\_count;

float gpa;

public:

Student(T Id, string Full, T Unique) : id(Id), full\_name(Full), unique\_identity(Unique), course\_count(0), gpa(0.0f)

{}

void calculate\_gpa() {

float sum = 0;

for (int i = 0; i < course\_count; ++i)

{

if (ptr[i]->grade == "A+" || ptr[i]->grade == "A" || ptr[i]->grade == "a+" || ptr[i]->grade == "a") {

sum += 4.0;

}

else if (ptr[i]->grade == "A-" || ptr[i]->grade == "a-")

{

sum += 3.67;

}

else if (ptr[i]->grade == "B+" || ptr[i]->grade == "b+")

{

sum += 3.33;

}

else if (ptr[i]->grade == "B" || ptr[i]->grade == "b")

{

sum += 3.0;

}

else if (ptr[i]->grade == "B-" || ptr[i]->grade == "b-")

{

sum += 2.67;

}

else if (ptr[i]->grade == "C+" || ptr[i]->grade == "c+")

{

sum += 2.33;

}

else if (ptr[i]->grade == "C" || ptr[i]->grade == "c")

{

sum += 2.0;

}

else if (ptr[i]->grade == "C-" || ptr[i]->grade == "c-")

{

sum += 1.67;

}

else if (ptr[i]->grade == "D+" || ptr[i]->grade == "d+")

{

sum += 1.33;

}

else if (ptr[i]->grade == "D" || ptr[i]->grade == "d")

{

sum += 1.0;

}

else if (ptr[i]->grade == "F" || ptr[i]->grade == "f")

{

sum += 0.0;

}

else

{

cout << "Invalid output "<<endl;

}

}

gpa = sum / course\_count;

}

void generate\_reports()

{

ofstream out("try.txt");

if (out.is\_open()) {

out << "The student name is : " << full\_name << '\n';

out << "The student id is : " << id << '\n';

out << "The student unique identity is : " << unique\_identity << '\n';

for (int i = 0; i < course\_count; ++i) {

out << "The course info at index " << i + 1 << " is :"<<endl;

out << "Course name is : " << ptr[i]->title << '\n';

out << "Course Grade is : " << ptr[i]->grade << '\n';

}

calculate\_gpa();

out << "Student Gpa is : "<<endl;

out << gpa << '\n';

}

}

~Student() {

for (int i = 0; i < course\_count; ++i) {

delete ptr[i];

}

}

void enrol\_course()

{

if (course\_count < 10 && course\_count >= 0) {

ptr[course\_count] = new Course<T>;

cout << "Enter data for course ."<<endl;

cout << "Enter the course title: "<<endl;

cin >> ptr[course\_count]->title;

cout << "Enter the course grade : "<<endl;

cin >> ptr[course\_count]->grade;

course\_count++;

}

}

void display()

{

cout << "The student name is : " << full\_name << '\n';

cout << "The student id is : " << id << '\n';

cout << "The student unique identity is : " << unique\_identity << '\n';

for (int i = 0; i < course\_count; ++i) {

cout << "The course info at index " << i + 1 << " is :"<<endl;

cout << "Course name is : " << ptr[i]->title << '\n';

cout << "Course Grade is : " << ptr[i]->grade << '\n';

}

calculate\_gpa();

cout << "Student Gpa is : "<<endl;

cout << gpa << '\n';

}

friend class Course<T>;

};

template<typename T>

class Course

{

private:

Student<T>\* Roaster[30];

string title;

int student\_count;

T grade;

public:

Course(string inpu = "OOP") : student\_count(0), title(inpu)

{}

void Add\_student() {

if (student\_count < 30 && student\_count >= 0)

{

Roaster[student\_count] = new Student<T>("23F\_000", "Naveed", "121");

cout << "Enter the student details that i am adding to course . "<<endl;

cout << "Enter the student full name : "<<endl;

cin >> Roaster[student\_count]->full\_name;

cout << "Enter the id : "<<endl;

cin >> Roaster[student\_count]->id;

cout << "Enter the unique id for a person : "<<endl;

cin >> Roaster[student\_count]->unique\_identity;

++student\_count;

}

else

{

cout << "Students are full ."<<endl;

}

}

friend class Student<T>;

};

int main()

{

Course<string> course1("OOP");

Student<string> student1("23F\_0599", "Abdullah", "202");

Student<string> student2("23F\_0000", "xyz", "512");

course1.Add\_student();

course1.Add\_student();

course1.Add\_student();

student2.enrol\_course();

cout << "Details of student 1 ."<<endl;

student1.display();

cout << "Details of student 2."<<endl;

student2.display();

student1.generate\_reports();

student2.generate\_reports();

system("pause");

return 0;

}

****

**Question#6**

#include<iostream>

#include <string>

using namespace std;

template<typename t1>

t1 MAX(t1 a, t1 b) {

if (a > b) {

return a;

}

else {

return b;

}

}

template<typename t1>

t1 MIN(t1 a, t1 b) {

if (a < b) {

return a;

}

else {

return b;

}

}

template<typename t1>

t1 add(t1 a, t1 b) {

return a + b;

}

template<typename t1>

t1 sub(t1 a, t1 b) {

return a - b;

}

template<typename t1>

t1 mul(t1 a, t1 b) {

return a \* b;

}

template <typename T>

T calculateMedian(T arr[], int size)

{

if (size % 2 != 0)

{

return arr[(size / 2)];

}

else

{

return arr[(size / 2) - 1];

}

}

template<typename t>

t calculateavg(t arr[], int size) {

t sum = 0;

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

sum += arr[i];

}

return sum / size;

}

template<typename t>

class circle {

t r;

const float pi = 3.14;

public:

circle(t r) {

this->r = r;

}

t area() {

return (pi \* r \* r);

}

t perimeter() {

return (2 \* pi \* r);

}

};

template<typename t>

class rect {

t length;

t width;

public:

rect(t length, t width) {

this->length = length;

this->width = width;

}

t area() {

return length \* width;

}

t perimeter() {

return 2 \* (length + width);

}

};

template<class t>

class triangle {

t base;

t height;

t side3;

public:

triangle(t base, t height, t side) {

this->base = base;

this->height = height;

this->side3 = side;

}

t area() {

return 0.5 \* base \* height;

}

t perimeter() {

return base + height + side3;

}

};

int main() {

int choice;

bool exit = false;

while (!exit) {

cout << "Enter 1 for basic arithmetic:" << endl;

cout << "Enter 2 for statistical calculation:" << endl;

cout << "Enter 3 for geometrical calculation:" << endl;

cout << "Enter 4 to exit :" << endl;

cin >> choice;

if (choice == 1) {

int n1, n2;

cout << "Enter two int numbers :" << endl;

cin >> n1 >> n2;

cout << "Addition of two int numbers are :" << add(n1, n2) << endl;

cout << "Subtraction of two int numbers are :" << sub(n1, n2) << endl;

cout << "Multiplication of two int numbers are :" << mul(n1, n2) << endl;

cout << "Maximum of two int numbers are :" << MAX(n1, n2) << endl;

cout << "Minimum of two int numbers are :" << MIN(n1, n2) << endl;

float n3, n4;

cout << "Enter 2 float numbers :" << endl;

cin >> n3 >> n4;

cout << "Addition of two float numbers are :" << add(n4, n3) << endl;

cout << "Subtraction of two float numbers are :" << sub(n3, n4) << endl;

cout << "Multiplication of two float numbers are :" << mul(n4, n3) << endl;

cout << "Maximum of two float numbers are :" << MAX(n4, n3) << endl;

cout << "Minimum of two float numbers are :" << MIN(n4, n3) << endl;

}

if (choice == 2) {

int\* ptr;

int size;

cout << "Enter size for int array :" << endl;

cin >> size;

ptr = new int[size];

cout << "Enter elements of int array :" << endl;

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

{

cin >> ptr[i];

}

cout << "Average of int array: " << calculateavg(ptr, size) << endl;

cout << "Median of int array is: " << calculateMedian(ptr, size) << endl;

delete[] ptr;

ptr = nullptr;

float\* pionter = new float[size];

cout << "Enter elements of float array :" << endl;

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

cin >> pionter[i];

}

cout << "Average of float array: " << calculateavg(pionter, size) << endl;

cout << "Median of float array is: " << calculateMedian(pionter, size) << endl;

delete[] pionter;

pionter = nullptr;

}

else if (choice == 3) {

int r;

cout << "Enter the radius of circle :" << endl;

cin >> r;

circle<int> obj(r);

cout << "The area of circle is: " << obj.area() << endl;

cout << "The perimeter of circle is: " << obj.perimeter() << endl;

float len, wid;

cout << "Enter the length and width of rectangle :" << endl;

cin >> len >> wid;

rect<float> object(len, wid);

cout << "The area of rectangle is: " << object.area() << endl;

cout << "The perimeter of rectangle is: " << object.perimeter() << endl;

cout << "Enter three sides of triangle :" << endl;

double b1, b2, b3;

cin >> b1 >> b2 >> b3;

triangle<double> tri(b1, b2, b3);

cout << "The area of triangle is: " << tri.area() << endl;

cout << "The perimeter of triangle is: " << tri.perimeter() << endl;

}

else if (choice == 4)

{

exit = true;

}

else

{

cout << "Invalid input :" << std::endl;

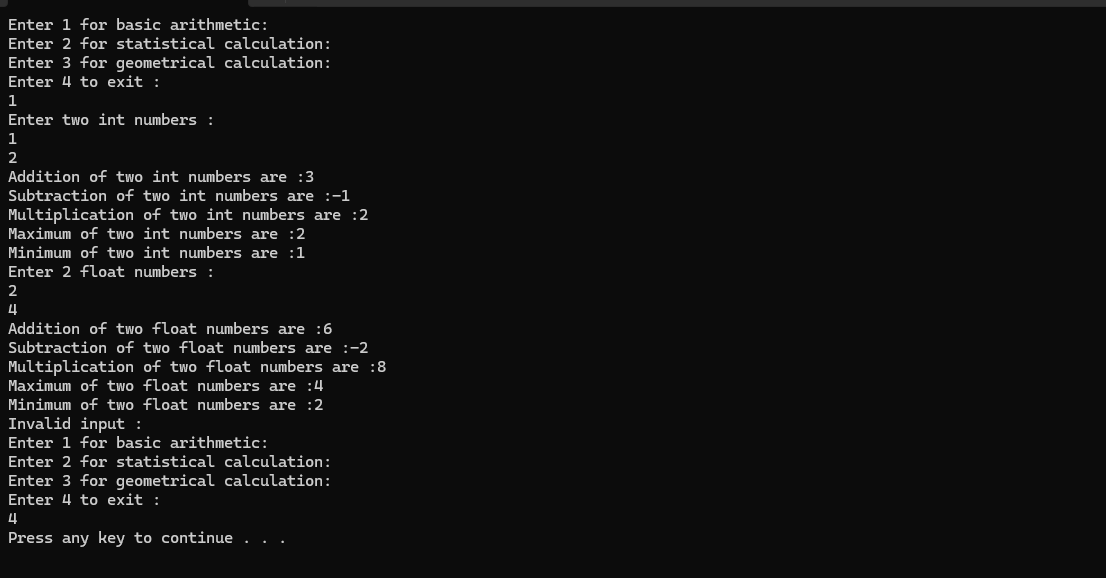
}

}

system("pause");

return 0;

}

****

**Question#7**

#include <iostream>

#include <string>

using namespace std;

class outofbounds

{

string message;

public:

outofbounds()

{

message = "Index out of bounds";

}

string what()

{

return message;

}

};

template<typename t>

class safearray

{

private:

int size;

t\* arr;

public:

safearray(int size = 0) : size(size)

{

arr = new t[size];

}

~safearray()

{

delete[] arr;

}

void input()

{

cout << "Enter " << size << " elements of array : \n";

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

{

cin >> arr[i];

}

cout << endl;

}

t& operator[](int index)

{

if (index >= size || index < 0)

{

throw outofbounds();

}

return arr[index];

}

friend ostream& operator<<(ostream& output, const safearray<t>& s)

{

for (int i = 0; i < s.size; i++)

{

output << s.arr[i] << " ";

}

return output;

}

};

int main()

{

int size;

cout << "Enter size = ";

cin >> size;

safearray<int> s(size);

s.input();

bool exit = false;

while (!exit)

{

try

{

int index;

cout << "Enter index whose information you want to get: ";

cin >> index;

cout << "value at index " << index << " = " << s[index] << endl;

exit = true;

}

catch (outofbounds& e)

{

cout << "Exception: " << e.what() << endl;

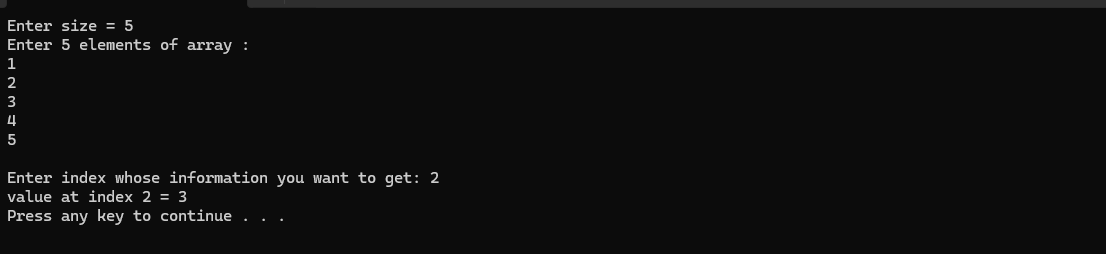
}

}

system("pause");

return 0;

}

****