|  |
| --- |
| **Assignment # 7**  ***Session*: Spring 2022 *Total marks*: 100**  ***Name*** : ***\_\_nimra maqbool\_\_ Roll number* : \_\_bsce21012\_\_\_** |

***Submission:***

• *Email instructor or TA if there are any questions. You cannot look at others’ solutions or use others’ solutions, however, you can discuss it with each other. Plagiarism will be dealt with according to the course policy.*

*• Submission after due time will not be accepted.*

**There should be a Report explaining your code and highlighting results. Follow this naming convention for your report RollNumber\_Assignment#.pdf e.g BSEE21001\_Assignment7.pdf.**

**TASKS**

**Q1** Make a class named as ‘TypeOfDay’. Identify the attribute/s and implement following functions:

void nextDay()

void previousDay()

Also make default and parametric constructors, setters and getters to implement functions .

|  |
| --- |
| **Function.cpp:**  void typeOfDay::setDate(int d) {  cout << "ENTER DATE = ";  cin >> d;  date = d; }  void typeOfDay::setMonth(int m) {  cout << "ENTER MONTH = ";  cin >> m;  month = m; } void typeOfDay::setYear(int y) {  cout << "ENTER YEAR = ";  cin >> y;  year = y; } int typeOfDay::getDate() {  return date; } int typeOfDay::getMonth() {  return month; } int typeOfDay::getYear() {  return year; } void typeOfDay::previousDay() {   if (month == 1) {  if (date == 1) {  month = 12;  date = 31;  year--;  cout << date << "-" << month << "-" << year << endl;  }  } else if (month == 1 || month == 3 || month == 5 || month == 7 || month == 8  || month == 10 || month == 12) {  if (date == 1) {  month--;  date = 31;  cout << date << "-" << month << "-" << year << endl;  } else {  date--;   cout << date << "-" << month << "-" << year << endl;  }  } else if (month == 2 || month == 4 || month == 6 || month == 9 || month == 11) {  if (date == 1) {  month--;  date = 30;  cout << date << "-" << month << "-" << year << endl;  } else {  date--;  cout << date << "-" << month << "-" << year << endl;  }  } }  void typeOfDay::nextDay() {  if (month == 12) {  if (date == 31) {  date = 1;  month = 1;  year = ++year;  cout << date << "-" << month << "-" << year << endl;  } else {  date++;  month = month;  year = ++year;  cout << date << "-" << month << "-" << year << endl;  }  } else if (month == 1 || month == 3 || month == 5 || month == 7 || month == 8  || month == 10) {  if (date == 31) {  date = 1;  month++;  year = ++year;  cout << date << "-" << month << "-" << year << endl;  } else {  date++;  month = month;  year = year;  cout << date << "-" << month << "-" << year << endl;  }  } else if (month == 2 || month == 4 || month == 6 || month == 9 || month == 11) {  if (date == 30) {  date = 1;  month++;  year = year;  cout << date << "-" << month << "-" << year << endl;  } else {  date++;  month = month;  year = year;  cout << date << "-" << month << "-" << year << endl;  }  }  }   * **In function.cpp I have defined the code of getter setter, next day and previous day function.** * **In setters I have set the value of month date and year.** * **In getters I have returned the value of month date and year.** * **In next day I have applied conditions to check that in which month this date lie and whether that month have 30 or 31 days etc.** * **In previous day I have applied conditions to check the month and if the month is January and date is 1 then it decrements the year also.**   **Function.h:**  class typeOfDay { private:  int month;  int date;  int year; public:  friend ostream &operator<<(ostream &ost,typeOfDay &tp);  typeOfDay() {  int d = 0;  int y = 0;  int m = 0;  date = d;  year = y;  month = m;   }   typeOfDay(int d, int y, int m) {  cout << “ENTER DATE = “;  cin >> d;  date = d;  cout << “ENTER YEAR = “;  cin >> y;  year = y;  cout << “ENTER MONTH = “;  cin >> m;  month = m;  }   void setDate(int d);   void setMonth(int m);   void setYear(int y);  int getDate();   int getMonth();   int getYear();   void nextDay();   void previousDay();   * **In function.cpp I have just declared the attributes and then made an default constructor and an parametrized constructor.** * **Then I declared the other getter setter and next and previous day function.** |

**Q2** In this task you will be using function overloading, by passing an integer to nextDay function and making it return today + nth day.

void nextday(int n)

|  |
| --- |
| **Function.h:**  class typeOfDay { private:  int month;  int date;  int year; public:  friend ostream &operator<<(ostream &ost,typeOfDay &tp);  typeOfDay() {  int d = 0;  int y = 0;  int m = 0;  date = d;  year = y;  month = m;   }   typeOfDay(int d, int y, int m) {  cout << "ENTER DATE = ";  cin >> d;  date = d;  cout << "ENTER YEAR = ";  cin >> y;  year = y;  cout << "ENTER MONTH = ";  cin >> m;  month = m;  }   void setDate(int d);   void setMonth(int m);   void setYear(int y);  int getDate();   int getMonth();   int getYear();   void nextDay();   void previousDay();   void nextDay(int n); };   * **In it I have just added the declaration of next day having parameter in it.**   **function.cpp:**  void typeOfDay::nextDay(int n) {  cout << "ENTER NUM OF DAYS YOU WANT TO ADD IN TODAY = ";  cin >> n;  if (month == 12) {  if (date == 31) {  date = 1;  date = date + (n-1);  month = 1;  year = ++year;  cout << date << "-" << month << "-" << year << endl;  } else {  date = date + n;  month = month;  year = ++year;  cout << date << "-" << month << "-" << year << endl;  }  } else if (month == 1 || month == 3 || month == 5 || month == 7 || month == 8  || month == 10) {  if (date == 31) {  date = 1;  date = date + (n-1);  month++;  year = ++year;  cout << date << "-" << month << "-" << year << endl;  } else {  date = date + n;  month = month;  year = year;  cout << date << "-" << month << "-" << year << endl;  }  } else if (month == 2 || month == 4 || month == 6 || month == 9 || month == 11) {  if (date == 30) {  date = 1;  date = date + (n-1);  month++;  year = year;  cout << date << "-" << month << "-" << year << endl;  } else {  date = date + n;  month = month;  year = year;  cout << date << "-" << month << "-" << year << endl;  }  } }   * **In this function I ask the user to enter a number and then add the number in the date and check that this date lie in which month and year.** |

**Q3** Now you are required to overload the ostream operator "<<" to print the current day, by declaring ostream as a friend class to the TypeOfDay class.

|  |
| --- |
| **Function.h:**  class typeOfDay { private:  int month;  int date;  int year; public:  friend ostream &operator<<(ostream &ost,typeOfDay &tp);  typeOfDay() {  int d = 0;  int y = 0;  int m = 0;  date = d;  year = y;  month = m;   }   typeOfDay(int d, int y, int m) {  cout << "ENTER DATE = ";  cin >> d;  date = d;  cout << "ENTER YEAR = ";  cin >> y;  year = y;  cout << "ENTER MONTH = ";  cin >> m;  month = m;  }   void setDate(int d);   void setMonth(int m);   void setYear(int y);  int getDate();   int getMonth();   int getYear();   void nextDay();   void previousDay();   void nextDay(int n); };   * **I have declared the friend function.**   **function.cpp:**  ostream &operator<<(ostream &ost ,typeOfDay &tp){  cout<<"TODAY DATE = ";  ost << tp.date<<"-"<<tp.month<<"-"<<tp.year<<endl;  return ost; }   * **I overloaded the << operator by using friend function defination.**   **output:**  **Text  Description automatically generatedText  Description automatically generated** |

**Q4** In this task you will define the class Complex with following data members;

double real;

double imaginary;

Define the default and parametric constructors and getter functions to get

the real and imaginary numbers.Also define the following additional functions:

printComplexNo();

addTwoComplexNo();

subtractTwoComplexNo();

multiplyTwoComplexNo();

divideTwoComplexNo();

getComplexConjugate();

And overload these operators:

+

-

\*

/

=

<<

>>

|  |
| --- |
| **Function.h:**  using namespace std;  class complex { private:  double real;  double imaginary; public:    complex() {  int r = 0;  int imag = 0;  real = r;  imaginary = imag;  cout << "DEFAULT CONSTRUCTOR IS CALLED." << endl;  }   complex(double r, double imag) {  cout << "ENTER REAL PART = ";  cin >> r;  cout << "ENTER IMAGINARY PART = ";  cin >> imag;  real = r;  imaginary = imag;  }   friend istream &operator>>(istream &ist, complex &c);   friend ostream &operator<<(ostream &ost, complex &c);   double getReal() ;   double getImaginary();   void printComplexNumber();   complex operator+(complex &c);   complex operator-(complex &c) ;   complex operator\*(complex &c);   complex operator/(const complex &c) const ;   void print();   void getConjugate();   void getComplexConjugate();   void addComplexNumber(complex C, complex C1);   void subComplexNumber(complex C, complex C1);   void multiplyComplexNumber(complex C, complex C1) ;   void divideComplexNumber(complex C, complex C1);   complex operator=(complex c);  };   * **In function.h I have declared the attributes needed and then I have made an default and parametrized constructor and declared other getter setter functions in it.**   **function.cpp:**  istream &operator>>(istream &ist, complex &c) {  cout << "ENTER REAL PART = ";  ist >> c.real;  cout << "ENTER IMAGINARY PART = ";  ist >> c.imaginary;  return ist; }  ostream &operator<<(ostream &ost, complex &c) {  ost << c.real;  ost << " + " << c.imaginary << "i" << endl;  return ost; } void complex::addComplexNumber(complex C, complex C1) {  real = C.real + C1.real;  imaginary = C.imaginary + C1.imaginary;  cout << "THE SUM = " << real << " + " << imaginary << "i" << endl; } void complex::subComplexNumber(complex C, complex C1) {  real = C.real - C1.real;  imaginary = C.imaginary - C1.imaginary;  if (imaginary < 0) {  cout << "SUBTRACTION = " << real << imaginary << "i" << endl;  } else {   cout << "SUBTRACTION = " << real << " + " << imaginary << "i" << endl;  } } void complex::multiplyComplexNumber(complex C, complex C1) {  real = C.real \* C1.real;  imaginary = C.imaginary \* C1.imaginary;  if (imaginary < 0) {  cout << "MULTIPLICATION = " << real << imaginary << "i" << endl;  } else {   cout << "MULTIPLICATION = " << real << " + " << imaginary << "i" << endl;  } } void complex::divideComplexNumber(complex C, complex C1) {  real = C.real / C1.real;  imaginary = C.imaginary / C1.imaginary;  if (imaginary < 0) {  cout << "MULTIPLICATION = " << real << imaginary << "i" << endl;  } else {   cout << "MULTIPLICATION = " << real << " + " << imaginary << "i" << endl;  } } void complex::print() {  cout << real << " + " << imaginary << "i" << endl; } complex complex::operator=(complex c) {  complex temp;  temp.real = c.real;  temp.imaginary = c.imaginary;  return temp; } complex complex::operator/(const complex &c) const {  complex temp;  temp.real = real / c.real;  temp.imaginary = imaginary / c.imaginary;  return temp; } complex complex::operator-(complex &c) {  complex temp;  temp.real = real - c.real;  temp.imaginary = imaginary - c.imaginary;  return temp; }  complex complex::operator\*(complex &c) {  complex temp;  temp.real = real \* c.real;  temp.imaginary = imaginary \* c.imaginary;  return temp; } complex complex::operator+(complex &c) {  complex temp;  temp.real = real + c.real;  temp.imaginary = imaginary + c.imaginary;  return temp; }   * **In main I have defined the setter of real and imaginary in which I have set the values of real and imaginary part.** * **Then I have made a function to add, subtract, multiply, divide and take conjugate of the 2 complex numbers.** * **Then I have done the addition subtraction multiplication and division by operator overloading.** * **I have overloaded the stream in, out and equal to operator to..**   **main.cpp:**  int main() {  double r;  double imag;  complex D;  int opt1; int opt; do{  cout<<"WHICH TASK DO YOU WANT TO PERFORM?"<<endl;  cout<<"1.TASK 1."<<endl;  cout<<"2.TASK 2."<<endl;  cout<<"3.EXIT."<<endl;  cin>>opt;  if(opt==1){  do{  cout<<"HOW DO YOU WANT TO PERFORM TASK 4?"<<endl;  cout<<"1.BY FUNCTION."<<endl;  cout<<"2.BY OPERATOR OVERLOADING."<<endl;  cin>>opt1;  if(opt1==1) {  do {  cout<<endl;  cout << "WHICH FUNCTION DO YOU WANT?" << endl;  cout << "1.CALL GETTERS AND PRINT COMPLEX NUMBERS." << endl;  cout << "2.SUM 2 COMPLEX NUMBERS." << endl;  cout << "3.SUBTRACT 2 COMPLEX NUMBERS." << endl;  cout << "4.MULTIPLY 2 COMPLEX NUMBERS." << endl;  cout << "5.DIVIDE 2 COMPLEX NUMBERS." << endl;  cout << "6.GET CONJUGATE OF COMPLEX NUMBER." << endl;  cout << "7.EXIT" << endl;  cin >> opt;  if (opt == 1) {  complex C(r, imag);  C.getReal();  C.getImaginary();  C.printComplexNumber();  }  if (opt == 2) {  complex c;  complex C(r, imag);  complex C1(r, imag);  c = C + C1;  c.addComplexNumber(C,C1);  }  if (opt == 3) {  complex c;  complex C(r, imag);  complex C1(r, imag);  c = C - C1;  c.subComplexNumber(C,C1);  }  if (opt == 4) {  complex c;  complex C(r, imag);  complex C1(r, imag);  c=C\*C1;  c.multiplyComplexNumber(C,C1);  }  if (opt == 5) {  complex c;  complex C(r, imag);  complex C1(r, imag);  c=C/C1;  c.divideComplexNumber(C,C1);  }  if (opt == 6) {  complex C(r, imag);  C.getConjugate();   }  if (opt == 7) {  cout<<"YOU CHOOSE TO EXIT..."<<endl;  exit(3);  }  } while (opt >= 1 && opt <= 7);  }  if(opt1==2){  do{  cout<<endl;  cout<<"WHICH FUNCTION DO YOU WANT?"<<endl;  cout<<"1.SUM 2 COMPLEX NUMBERS."<<endl;  cout<<"2.SUBTRACT 2 COMPLEX NUMBERS."<<endl;  cout<<"3.MULTIPLY 2 COMPLEX NUMBERS."<<endl;  cout<<"4.DIVIDE 2 COMPLEX NUMBERS."<<endl;  cout<<"5.ISTREAM OPERATOR OVERLOADING."<<endl;  cout<<"6.OSTREAM OPERATOR OVERLOADING."<<endl;  cout<<"7.EQUAL TO OPERATOR OVERLOADING."<<endl;  cout<<"8.EXIT"<<endl;  cin>>opt;  if(opt==1){  complex C(r, imag);  complex C1(r, imag);  complex c = C + C1;  c.print();  }  if(opt==2){  complex C(r, imag);  complex C1(r, imag);  complex c = C - C1;  c.print();  }  if(opt==3){  complex C(r, imag);  complex C1(r, imag);  complex c = C \* C1;  c.print();  }  if(opt==4){  complex C(r, imag);  complex C1(r, imag);  complex c = C / C1;  c.print();  }  if(opt==5){  std::cin >> D;  }  if(opt==6){  std::cout << "COMPLEX NUMBER = ";  std::cout << D;  }  if(opt==7){  complex C(r, imag);  complex C1(r, imag);  complex c=(C=C1);  c.print();   }  if(opt==8){  cout<<"YOU CHOOSE TO EXIT..."<<endl;  exit(3);  }  }while(opt>=1 && opt<=8);  }  if(opt1==3){  cout<<"YOU CHOOSE TO EXIT..."<<endl;  exit(3);  }  }while(opt1>=1 && opt1<=3);  }  if(opt==2){  do {  cout << "ENTER NUMBERS TO RUN THE TASK." << endl;  cout << "1.DEFAULT CONSTRUCTOR AND PARAMETRIZED CONSTRUCTOR." << endl;  cout << "2.SET VALUES AND GET VALUES." << endl;  cout << "3.NEXT DAY." << endl;  cout << "4.PREVIOUS DAY." << endl;  cout << "5.TODAY DATE BY OPERATOR OVERLOADING." << endl;  cout << "6.EXIT" << endl;  cin >> opt;  if (opt == 1) {  int d;  int y;  int m;  typeOfDay TP(d, y, m);  }  if (opt == 2) {  int d;  int y;  int m;  typeOfDay TP;  TP.setDate(d);  TP.setMonth(m);  TP.setYear(y);  TP.getDate();  TP.getMonth();  TP.getYear();  }  if (opt == 4) {  int d;  int y;  int m;  typeOfDay TP(d, y, m);  TP.previousDay();  }  if (opt == 3) {  cout<<"1.WITHOUT OVERLOADING."<<endl;  cout<<"2.WITH OVERLOADING."<<endl;  cout<<"3.EXIT."<<endl;  cin>>opt;  if(opt==1){   int d;  int y;  int m;  typeOfDay Tp;  typeOfDay TP(d, y, m);  TP.nextDay();  }  if(opt==2){  int d;  int y;  int m;  int n;  typeOfDay Tp;  typeOfDay TP(d, y, m);  TP.nextDay(n);  }   }  if(opt==5){  int d;  int y;  int m;  int n;  typeOfDay Tp;  typeOfDay TP(d, y, m);  std::cout<<TP;  }  if (opt == 6) {  cout << "YOU CHOOSE TO EXIT.." << endl;  exit(3);  }   }while(opt>=1 && opt<=5);   }  if(opt==3){  cout<<"YOU CHOOSE TO EXIT.."<<endl;  exit(4);  } }while(opt>=1 && opt<=3);  return 0; }   * **In main.cpp I have made a menu for arranged calling by the will of user.**   **output:**  **Text  Description automatically generatedText  Description automatically generatedText  Description automatically generatedText  Description automatically generated with medium confidenceA picture containing graphical user interface  Description automatically generatedText  Description automatically generated** |