QUESTION 1:

#include<iostream>

#include<string>

using namespace std;

void permute(string a, int c, int r)

{

if (c != r)

{

for (int i = c; i <= r; i++)

{

swap(a[c], a[i]); //built-in swap function

permute(a, c + 1, r); //recursive calling

swap(a[c], a[i]);

}

}

else

{

cout << a << endl;

}

}

int main()

{

string str;

cout << "Enter your string : ";

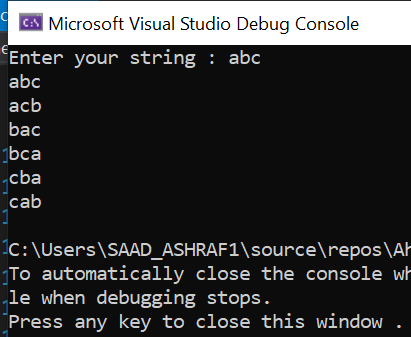
cin >> str;

int n = str.size();

permute(str, 0, n - 1); //calling in main

return 0;

}



QUESTION 3:

#include<iostream>

#include<string>

#include<ctime>

using namespace std;

struct timer

{

int secs = 0, mint = 0, hours = 0;

};

class testing

{

int test\_id;

string question; static int count;

public:

timer obj1;

void setTestId(int test\_id)

{

this->test\_id = test\_id;

}

void setQuestion(string question)

{

this->question = question;

}

void setTimer(timer obj)

{

obj.mint = rand() % 60;

obj.hours = rand() % 12;

obj.secs = rand() % 280;

obj1.hours = obj.hours;

this->obj1.mint = obj.mint;

this->obj1.secs = obj.secs;

}

void display()

{

if (count == 0)

{

cout << "Tid Question time" << endl;

cout << test\_id << " " << question << " " << test\_id - 1 << "\* ";

cout << obj1.hours << ":" << obj1.mint << ":" << obj1.secs << endl; count++;

}

else

{

cout << test\_id << " " << question << " " << test\_id - 1 << "\* ";

cout << obj1.hours << ":" << obj1.mint << ":" << obj1.secs << endl;

}

}

};

int testing::count = 0;

int main()

{

int test = 0;

string question;

testing a[10];

timer var;

var.hours = 12;

var.mint = 10;

var.secs = 1200;

for (int i = 0; i < 10; i++){

a[i].setTestId(i + 1);

a[i].setQuestion("This is question no.");

a[i].setTimer(var);

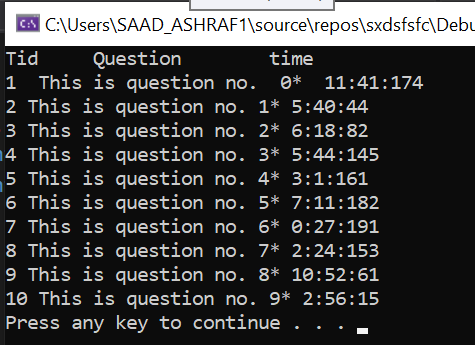
a[i].display();

}

system("pause");

return 0;

}



QUESTION 4:

#include<iostream>

using namespace std;

class Example {

int data;

static int count;

public:

Example(int y = 10) :data(y) {

}

int getIncremenetData() const { //const error//const error

return ++data; //data is not initalized so it can not be incremented

}

static int getcount() {

cout << "Data is" << data << endl; //static function for static variables return count;

return count;

}

};

//not iniatialze the static value outuside the class

int Example::count = 0;

int main() {

}

QUESTION 5:

#include <iostream>

#include <iomanip>

using namespace std;

class Account {

public:

Account(double);

void credit(double);

bool debit(double);

void setBalance(double);

double getBalance();

private:

double balance;

};

class SavingsAccount : public Account {

public:

SavingsAccount(double, double);

double calculateInterest();

private:

double interestRate;

};

class CheckingAccount : public Account {

public:

CheckingAccount(double, double);

void credit(double);

bool debit(double);

private:

double transactionFee;

void chargeFee();

};

Account::Account(double initialBalance) {

if (initialBalance >= 0.0)

balance = initialBalance;

else

throw invalid\_argument("Initial balance can never be negative");

}

void Account::credit(double amount) {

balance = balance + amount;

}

bool Account::debit(double amount) {

if (amount < balance) {

balance = balance - amount;

return true;

}

else

{

cout << "Debit amount exceeded account balance." << endl;

return false;

}

}

void Account::setBalance(double newBalance) {

balance = newBalance;

}

double Account::getBalance() {

return balance;

}

SavingsAccount::SavingsAccount(double initialBalance, double rate) : Account(initialBalance) // initialize base class

{

if (rate >= 0.0)

interestRate = rate;

else

throw invalid\_argument("Interest rate should be >= 0.0");

}

double SavingsAccount::calculateInterest() {

return getBalance() \* interestRate;

}

CheckingAccount::CheckingAccount(double initialBalance, double fee) : Account(initialBalance) {

if (fee >= 0.0)

transactionFee = fee;

else

throw invalid\_argument("Transaction fee should be >= 0.0");

}

void CheckingAccount::credit(double amount) {

Account::credit(amount);

chargeFee();

}

bool CheckingAccount::debit(double amount) {

bool success = Account::debit(amount);

if (success)

{

chargeFee();

return true;

}

else

return false;

}

void CheckingAccount::chargeFee() {

Account::setBalance(getBalance() - transactionFee);

cout << "$" << transactionFee << " transaction fee has been charged." << endl;

}

int main() {

int a1, a2, a3, a4, a5, a6;

cout << "enter balance of account 1: ";

cin >> a1;

cout << "enter balance of account 2: ";

cin >> a2;

cout << "enter balance of account 3: ";

cin >> a3;

cout << "How much do you want to debit from account1: ";

cin >> a4;

cout << "How much do you want to debit from account2: ";

cin >> a5;

cout << "How much do you want to debit from account3: ";

cin >> a6;

system("pause");

system("cls");

Account account1(a1);

SavingsAccount account2(a2, .03);

CheckingAccount account3(a3, 1.0);

cout << fixed << setprecision(2);

cout << "account1 balance is : $" << account1.getBalance() << endl;

cout << "account2 balance is : $" << account2.getBalance() << endl;

cout << "account3 balance is : $" << account3.getBalance() << endl;

cout << "\nAttempting to debit $" << a4 << " from account1." << endl;

account1.debit(a4);

cout << "\nAttempting to debit $" << a5 << " from account2." << endl;

account2.debit(a5);

cout << "\nAttempting to debit $" << a6 << " from account3." << endl;

account3.debit(a6);

cout << "\naccount1 balance: $" << account1.getBalance() << endl;

cout << "account2 balance: $" << account2.getBalance() << endl;

cout << "account3 balance: $" << account3.getBalance() << endl;

cout << "\nCrediting $90.00 to account1." << endl;

account1.credit(90.0);

cout << "\nCrediting $25.00 to account2." << endl;

account2.credit(25.0);

cout << "\nCrediting $10.00 to account3." << endl;

account3.credit(10.0);

cout << "\naccount1 balance: $" << account1.getBalance() << endl;

cout << "account2 balance: $" << account2.getBalance() << endl;

cout << "account3 balance: $" << account3.getBalance() << endl;

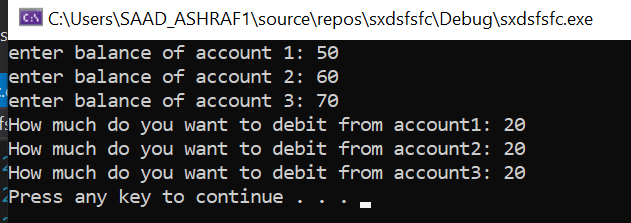
double interestEarned = account2.calculateInterest();

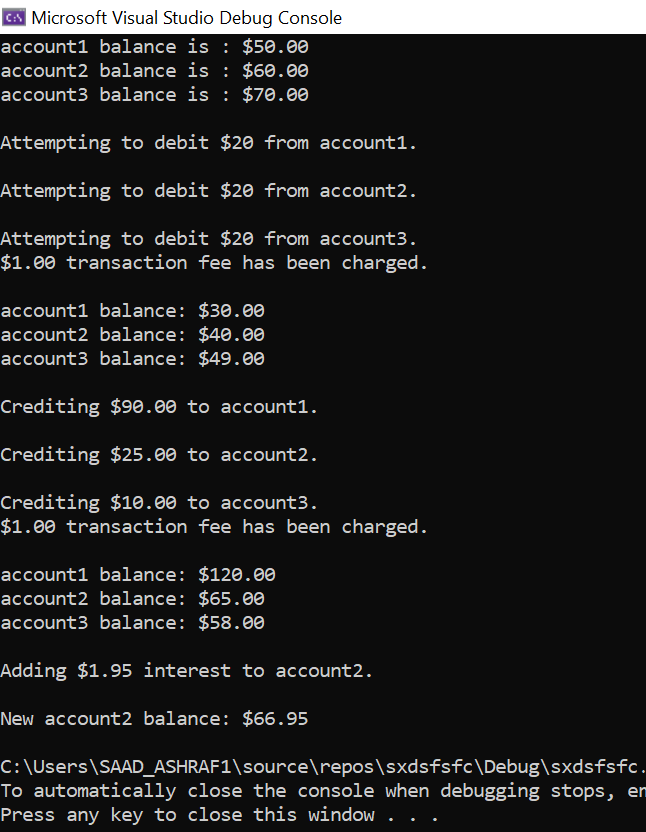
cout << "\nAdding $" << interestEarned << " interest to account2." << endl;

account2.credit(interestEarned);

cout << "\nNew account2 balance: $" << account2.getBalance() << endl;

}





QUESTION NO 6:

#include <iostream>

#include <string>

#include <iomanip>

using namespace std;

class Package

{

private:

string sender\_name;

string sender\_address;

string sender\_city;

string sender\_state;

string sender\_ZIP;

string recipient\_name;

string recipient\_address;

string recipient\_city;

string recipient\_state;

string recipient\_ZIP;

double weight;

double costperounce;

public:

Package(string sender\_n, string sender\_addr, string sender\_c,

string sender\_s, string sender\_Z, string recipient\_n, string recipient\_addr,

string recipient\_c, string recipient\_s, string recipient\_Z, double wei,

double cost);

void setsender\_name(string sender\_n);

string displaysender\_name();

void setsender\_address(string sender\_addr);

string displaysender\_address();

void setsender\_city(string sender\_c);

string displaysender\_city();

void setsender\_state(string sender\_s);

string displaysender\_state();

void setsender\_ZIP(string sender\_Z);

string displaysender\_zip();

void setrecipient\_name(string recipient\_n);

string displayrecipient\_name();

void setrecipient\_address(string recipient\_addr);

string getrecipient\_address();

void setrecipient\_city(string recipient\_c);

string displayrecipient\_city();

void setrecipient\_state(string recipient\_s);

string displayrecipient\_state();

void setrecipient\_ZIP(string recipient\_Z);

string displayrecipient\_zip();

void setweight(double w);

double getweight();

void setcostperounce(double cost);

double getcostperounce();

double costcalculator();

};

Package::Package(string sender\_n, string sender\_addr, string sender\_c, string

sender\_s, string sender\_Z, string recipient\_n, string recipient\_addr, string

recipient\_c, string recipient\_s, string recipient\_Z, double wei, double cost)

{

sender\_name = sender\_n;

sender\_address = sender\_addr;

sender\_city = sender\_c;

sender\_state = sender\_s;

sender\_ZIP = sender\_Z;

recipient\_name = recipient\_n;

recipient\_address = recipient\_addr;

recipient\_city = recipient\_c;

recipient\_state = recipient\_s;

recipient\_ZIP = recipient\_Z;

if (wei > 0.0 && cost > 0.0)

{

weight = wei;

costperounce = cost;

}

else

{

weight = 0.0;

costperounce = 0.0;

}

}

void Package::setsender\_name(string sender\_n)

{

this-> sender\_name = sender\_n;

}

string Package::displaysender\_name()

{

return sender\_name;

}

void Package::setsender\_address(string sender\_addr)

{

this-> sender\_address = sender\_addr;

}

string Package::displaysender\_address()

{

return sender\_address;

}

void Package::setsender\_city(string sender\_c)

{

sender\_city = sender\_c;

}

string Package::displaysender\_city()

{

return sender\_city;

}

void Package::setsender\_state(string sender\_s)

{

this-> sender\_state = sender\_s;

}

string Package::displaysender\_state()

{

return sender\_state;

}

void Package::setsender\_ZIP(string sender\_Z)

{

this-> sender\_ZIP = sender\_Z;

}

string Package::displaysender\_zip()

{

return sender\_ZIP;

}

void Package::setrecipient\_name(string recipient\_n)

{

this-> recipient\_name = recipient\_n;

}

string Package::displayrecipient\_name()

{

return recipient\_name;

}

void Package::setrecipient\_address(string recipient\_addr)

{

this-> recipient\_address = recipient\_addr;

}

string Package::getrecipient\_address()

{

return recipient\_address;

}

void Package::setrecipient\_city(string recipient\_c)

{

this-> recipient\_city = recipient\_c;

}

string Package::displayrecipient\_city()

{

return recipient\_city;

}

void Package::setrecipient\_state(string recipient\_s)

{

this-> recipient\_state = recipient\_s;

}

string Package::displayrecipient\_state()

{

return recipient\_state;

}

void Package::setrecipient\_ZIP(string recipient\_Z)

{

this-> recipient\_ZIP = recipient\_Z;

}

string Package::displayrecipient\_zip()

{

return recipient\_ZIP;

}

void Package::setweight(double w)

{

weight = (w < 0.0) ? 0.0 : w;

}

double Package::getweight()

{

return weight;

}

void Package::setcostperounce(double cost)

{

costperounce = (cost < 0.0) ? 0.0 : cost;

}

double Package::getcostperounce()

{

return costperounce;

}

double Package::costcalculator()

{

double result;

result = weight \* costperounce;

return result;

}

class TwoDayPackage : public Package

{

private:

double two\_day\_delivery\_fee;

public:

TwoDayPackage(string sender\_n, string sender\_addr, string

sender\_c, string sender\_s, string sender\_Z, string recipient\_n,

string recipient\_addr, string recipient\_c, string recipient\_s,

string recipient\_Z, double wei, double cost, double delivery\_fee);

double gettwo\_day\_delivery\_fee();

void settwo\_day\_delivery\_fee(double delivery\_fee);

double costcalculator();

};

TwoDayPackage::TwoDayPackage(string sender\_n, string sender\_addr,

string sender\_c, string sender\_s, string sender\_Z, string recipient\_n,

string recipient\_addr, string recipient\_c, string recipient\_s,

string recipient\_Z, double wei, double cost, double delivery\_fee)

:Package(sender\_n, sender\_addr, sender\_c, sender\_s, sender\_Z, recipient\_n,

recipient\_addr, recipient\_c, recipient\_s, recipient\_Z, wei, cost)

{

settwo\_day\_delivery\_fee(delivery\_fee);

}

double TwoDayPackage::gettwo\_day\_delivery\_fee()

{

return two\_day\_delivery\_fee;

}

void TwoDayPackage::settwo\_day\_delivery\_fee(double delivery\_fee)

{

two\_day\_delivery\_fee = delivery\_fee;

}

double TwoDayPackage::costcalculator()

{

double result;

result = Package::costcalculator() + two\_day\_delivery\_fee;

return result;

}

class OvernightPackage : public Package

{

private:

double overnight\_delivery\_fee;

public:

OvernightPackage(string sender\_n, string sender\_addr, string sender\_c,

string sender\_s, string sender\_Z, string recipient\_n, string recipient\_addr,

string recipient\_c, string recipient\_s, string recipient\_Z, double wei,

double cost, double delivery\_fee);

double costcalculator();

double getovernight\_delivery\_fee();

void setovernight\_delivery\_fee(double delivery\_fee);

};

OvernightPackage::OvernightPackage(string sender\_n, string sender\_addr,

string sender\_c, string sender\_s, string sender\_Z, string recipient\_n,

string recipient\_addr, string recipient\_c, string recipient\_s,

string recipient\_Z, double wei, double cost, double delivery\_fee)

:Package(sender\_n, sender\_addr, sender\_c, sender\_s, sender\_Z, recipient\_n,

recipient\_addr, recipient\_c, recipient\_s, recipient\_Z, wei, cost)

{

setovernight\_delivery\_fee(delivery\_fee);

}

double OvernightPackage::getovernight\_delivery\_fee()

{

return overnight\_delivery\_fee;

}

void OvernightPackage::setovernight\_delivery\_fee(double delivery\_fee)

{

overnight\_delivery\_fee = delivery\_fee;

}

double OvernightPackage::costcalculator()

{

double result;

result = (getcostperounce() + overnight\_delivery\_fee) \* getweight();

return result;

}

int main(int argc, char\* argv[])

{

string a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t;

cout << "FOR OVEr-NIGHT PACKAGE" << endl;

cout << "ENTER SENDER ADDRESS : " << endl;

cout << "Enter name : ";

cin >> a;

cout << "Enter address : ";

cin >> b;

cout << "Enter city : ";

cin >> c;

cout << "Enter state : ";

cin >> d;

cout << "Enter zip code : ";

cin >> e;

cout << "ENTER RECIPITENT ADDRESS : " << endl;

cout << "Enter name : ";

cin >> f;

cout << "Enter address : ";

cin >> g;

cout << "Enter city : ";

cin >> h;

cout << "Enter state : ";

cin >> i;

cout << "Enter zip code : ";

cin >> j;

system("pause");

system("cls");

cout << "FOR TWO-DAY PACKAGE" << endl;

cout << "ENTER SENDER ADDRESS : " << endl;

cout << "Enter name : ";

cin >> k;

cout << "Enter address : ";

cin >> l;

cout << "Enter city : ";

cin >> m;

cout << "Enter state : ";

cin >> n;

cout << "Enter zip code : ";

cin >> o;

cout << "ENTER RECIPITENT ADDRESS : " << endl;

cout << "Enter name : ";

cin >> p;

cout << "Enter address : ";

cin >> q;

cout << "Enter city : ";

cin >> r;

cout << "Enter state : ";

cin >> s;

cout << "Enter zip code : ";

cin >> t;

system("pause");

system("cls");

OvernightPackage overnight(a, b, c, d, e, f, g, h, i, j, 15.00, 2.50, 3.00);

TwoDayPackage twoday( k, l, m, n, o, p, q, r, s,t, 20.00, 1.00, 10.00);

cout << "Overnight Delivery\n";

cout << "Sender : " << endl;;

cout << overnight.displaysender\_name() << endl;

cout << overnight.displaysender\_address() <<endl;

cout << overnight.displaysender\_city() << endl;

cout << overnight.displaysender\_state() << endl;

cout << overnight.displaysender\_zip() << endl;

cout << endl<<endl;

cout << "Recipient :" << endl;;

cout << overnight.displayrecipient\_name() <<endl;

cout << overnight.displaysender\_address() << endl;

cout << overnight.displayrecipient\_city() << endl;

cout << overnight.displayrecipient\_state() << endl;

cout<< overnight.displayrecipient\_zip() << endl;

cout << "Cost is : $ " << overnight.costcalculator() <<endl;

cout <<endl<<endl<<endl;

cout << "2 Day Delivery Pacakge :" << endl;

cout << "Sender : " << endl;

cout << twoday.displaysender\_name() << endl;

cout << twoday.displaysender\_address() << endl;

cout << twoday.displaysender\_city() << endl;

cout << twoday.displaysender\_state() << endl;

cout << twoday.displaysender\_zip() << endl;

cout << endl << endl;

cout << "Recipient :" << endl;;

cout << twoday.displayrecipient\_name() << endl;

cout << twoday.displaysender\_address() << endl;

cout << twoday.displayrecipient\_city() << endl;

cout << twoday.displayrecipient\_state() << endl;

cout << twoday.displayrecipient\_zip() << endl;

cout << "Cost is : $ " << twoday.costcalculator() << endl;

system("pause");

return 0;

}

