**Question 5 :** Let we have to prepare the final result of each student for a particular subject. The final marks are sum of marks obtained by the student in T1, T2, T3, P1, P2, and attendance in theory class. Let there are following classes:

(a) Student: Its data members are student name and roll number and member function is to print the values of the data members.

(b) T1T2T3: Its data members are marks obtained by a student in T1, T2, and T3 and member function is to print the values of the data members.

(c) P1P2: Its data members are marks obtained by a student in P1 and P2 and member function is to print the values of the data members.

(d) Attendance: Data member of this class is the percentage of attendance of a student in the theory class and member function is to print the values of the data member.

(e) Total: Data members of this class are total marks obtained and the grade secured by a student and member function is to print the values of the data members.Apart from that, there are two other classes:

(a) Faculty: which have no data member but have a member function to assign the marks of T1, T2, T3, P1, P2, and percentage of attendance to each student.

(b) Administration: which have no data member but have member functions to enter the name and roll number of each student, and to calculate the total marks and final grade of each student. Grade ‘A’ for> 80% marks, ‘B’ for 70 to 80%, ‘C’ for 60 to 70%, D for 50 to 60%, F for <50%.Base on above information, do the following:

(i) Write a function which prints the name and grades of all students in the ascending order of the grades. In case of same grade, print all respective names in alphabetical order.

(ii) Write a function which prints the name and grades of all students in the alphabetical order of the name of the students.

(iii) Write a function to search the grade of a student based on the first name of the student. In case of multiple entries with same name, print all the names with roll number and respective grades.

Here, consider Student as a base class whose derived classes are T1T2T3, P1P2, and Attendance. Further, consider the class Total which inherits the classes T1T2T3, P1P2, and Attendance as multiple inheritance. Classes Faculty and Administration are independent classes.

**Code:**

#include <iostream>

using std::cout;

using std::endl;

using std::string;

class student

{

protected:

string name;

int roll\_no;

public:

void display()

{

cout << "Name : " << this->name << " Roll no : " << roll\_no << endl;

}

};

class T1 : virtual public student

{

protected:

int sub\_marks[4];

public:

void display\_marks()

{

cout << " sub1 " << sub\_marks[0] << " sub2 " << sub\_marks[1] << " sub3 " << sub\_marks[2] << " sub4 " << sub\_marks[3] << endl;

}

};

class T2 : virtual public student

{

protected:

int sub\_marks[4];

public:

void display\_marks()

{

cout << " sub1 " << sub\_marks[0] << " sub2 " << sub\_marks[1] << " sub3 " << sub\_marks[2] << " sub4 " << sub\_marks[3] << endl;

}

};

class T3 : virtual public student

{

protected:

int sub\_marks[4];

public:

void display\_marks()

{

cout << " sub1 " << sub\_marks[0] << " sub2 " << sub\_marks[1] << " sub3 " << sub\_marks[2] << " sub4 " << sub\_marks[3] << endl;

}

};

class P1 : virtual public student

{

protected:

int sub\_marks[4];

public:

void display\_marks()

{

cout << " sub1 " << sub\_marks[0] << " sub2 " << sub\_marks[1] << " sub3 " << sub\_marks[2] << " sub4 " << sub\_marks[3] << endl;

}

};

class P2 : virtual public student

{

protected:

int sub\_marks[4];

public:

void display\_marks()

{

cout << " sub1 " << sub\_marks[0] << " sub2 " << sub\_marks[1] << " sub3 " << sub\_marks[2] << " sub4 " << sub\_marks[3] << endl;

}

};

class Attendance : virtual public student

{

protected:

int attendance\_percent;

public:

void display()

{

cout << "Attendance pecentage is " << attendance\_percent << endl;

}

};

class Total : public T1, public T2, public T3, public P1, public P2, public Attendance

{

int total\_marks[4] = {0};

int total\_percent;

public:

void total\_mark()

{

int Attendance\_mark;

if (attendance\_percent >= 90)

{

Attendance\_mark = 5;

}

else if (attendance\_percent >= 88)

{

Attendance\_mark = 4;

}

else if (attendance\_percent >= 86)

{

Attendance\_mark = 3;

}

else if (attendance\_percent >= 83)

{

Attendance\_mark = 2;

}

else if (attendance\_percent >= 80)

{

Attendance\_mark = 1;

}

else

{

Attendance\_mark = 0;

}

for (int k = 0; k < 4; k++)

{

total\_marks[k] = T1::sub\_marks[k] + T2::sub\_marks[k] + T3::sub\_marks[k] + P1::sub\_marks[k] + P2::sub\_marks[k] + Attendance\_mark;

}

total\_percent = (total\_marks[0] + total\_marks[1] + total\_marks[2] + total\_marks[3]) / 4;

}

friend void studgrade(Total \*, int);

friend void studgrade\_lex(Total \*, int);

friend void search(Total \*, string, int);

friend class Faculty;

friend class Administration;

};

void studgrade(Total \*ptr, int size)

{

cout << endl

<< "------------" << endl;

string gradetemp[size][2];

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

{

ptr[i].total\_mark();

if (ptr[i].total\_percent > 80)

{

gradetemp[i][0] = "A";

}

else if (ptr[i].total\_percent > 70)

{

gradetemp[i][0] = "B";

}

else if (ptr[i].total\_percent > 60)

{

gradetemp[i][0] = "C";

}

else if (ptr[i].total\_percent > 50)

{

gradetemp[i][0] = "D";

}

else if (ptr[i].total\_percent <= 50)

{

gradetemp[i][0] = "F";

}

gradetemp[i][1] = ptr[i].name;

//cout << gradetemp[i][0] << "\t" << gradetemp[i][1] << endl;

}

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

{

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

{

if (gradetemp[j][0] < gradetemp[j + 1][0])

{

string temp1, temp2;

temp1 = gradetemp[j][0];

temp2 = gradetemp[j][1];

gradetemp[j][0] = gradetemp[j + 1][0];

gradetemp[j][1] = gradetemp[j + 1][1];

gradetemp[j + 1][0] = temp1;

gradetemp[j + 1][1] = temp2;

}

}

//cout << gradetemp[i][0] << "\t" << gradetemp[i][1] << endl;

}

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

{

for (int j = 0; j < size - i - 1; ++j)

{

if (gradetemp[j][0] == gradetemp[j + 1][0] && gradetemp[j][1] < gradetemp[j + 1][1])

{

string temp1, temp2;

temp1 = gradetemp[j][0];

temp2 = gradetemp[j][1];

gradetemp[j][0] = gradetemp[j + 1][0];

gradetemp[j][1] = gradetemp[j + 1][1];

gradetemp[j + 1][0] = temp1;

gradetemp[j + 1][1] = temp2;

}

}

}

for (int i = size-1; i >= 0; i--)

{

//cout << "a" <<endl;

cout << gradetemp[i][0] << "\t" << gradetemp[i][1] << endl;

}

cout << endl

<< "------------" << endl;

}

void studgrade\_lex(Total \*ptr, int size)

{

string gradetemp[size][2];

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

{

if (ptr[i].total\_percent > 80)

{

gradetemp[i][0] = "A";

}

else if (ptr[i].total\_percent > 70)

{

gradetemp[i][0] = "B";

}

else if (ptr[i].total\_percent > 60)

{

gradetemp[i][0] = "C";

}

else if (ptr[i].total\_percent > 50)

{

gradetemp[i][0] = "D";

}

else if (ptr[i].total\_percent <= 50)

{

gradetemp[i][0] = "F";

}

gradetemp[i][1] = ptr[i].name;

}

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

{

for (int j = 0; j < size - i - 1; ++j)

{

if (gradetemp[j][1] > gradetemp[j + 1][1])

{

string temp1, temp2;

temp1 = gradetemp[j][0];

temp2 = gradetemp[j][1];

gradetemp[j][0] = gradetemp[j + 1][0];

gradetemp[j][1] = gradetemp[j + 1][1];

gradetemp[j + 1][0] = temp1;

gradetemp[j + 1][1] = temp2;

}

}

}

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

{

cout << gradetemp[i][0] << "\t" << gradetemp[i][1] << endl;

}

cout << endl

<< "------------" << endl;

}

void search(Total \*ptr, string temp, int size)

{

int space\_index\_temp, space\_index;

for (int i = 0; i < temp.length(); i++)

{

if (temp.at(i) == ' ')

{

//cout << i <<endl;

space\_index\_temp = i;

break;

}

}

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

{

string temproary = ptr[k].name;

for (int i = 0; i < temproary.length(); i++)

{

if (temproary.at(i) == ' ')

{

//cout << i <<endl;

space\_index = i;

break;

}

}

if (!(temproary.compare(0, space\_index, temp, 0, space\_index\_temp)))

{

if (ptr[k].total\_percent > 80)

{

cout << "A\t";

}

else if (ptr[k].total\_percent > 70)

{

cout << "B\t";

}

else if (ptr[k].total\_percent > 60)

{

cout << "C\t";

}

else if (ptr[k].total\_percent > 50)

{

cout << "D\t";

}

else if (ptr[k].total\_percent <= 50)

{

cout << "F\t";

}

ptr[k].student::display();

}

}

cout << endl

<< "------------" << endl;

}

class Faculty

{

public:

void addresult(Total &obj, int T1\_1, int T1\_2, int T1\_3, int T1\_4, int T2\_1, int T2\_2, int T2\_3, int T2\_4, int T3\_1, int T3\_2, int T3\_3, int T3\_4, int P1\_1, int P1\_2, int P1\_3, int P1\_4, int P2\_1, int P2\_2, int P2\_3, int P2\_4, int Attend)

{

obj.T1::sub\_marks[0] = T1\_1;

obj.T1::sub\_marks[1] = T1\_2;

obj.T1::sub\_marks[2] = T1\_3;

obj.T1::sub\_marks[3] = T1\_4;

obj.T2::sub\_marks[0] = T2\_1;

obj.T2::sub\_marks[1] = T2\_2;

obj.T2::sub\_marks[2] = T2\_3;

obj.T2::sub\_marks[3] = T2\_4;

obj.T3::sub\_marks[0] = T3\_1;

obj.T3::sub\_marks[1] = T3\_2;

obj.T3::sub\_marks[2] = T3\_3;

obj.T3::sub\_marks[3] = T3\_4;

obj.P1::sub\_marks[0] = P1\_1;

obj.P1::sub\_marks[1] = P1\_2;

obj.P1::sub\_marks[2] = P1\_3;

obj.P1::sub\_marks[3] = P1\_4;

obj.P2::sub\_marks[0] = P2\_1;

obj.P2::sub\_marks[1] = P2\_2;

obj.P2::sub\_marks[2] = P2\_3;

obj.P2::sub\_marks[3] = P2\_4;

obj.attendance\_percent = Attend;

}

};

class Administration

{

public:

void addstudentdetail(Total &obj, int roll, string name)

{

obj.name = name;

obj.roll\_no = roll;

}

};

int main()

{

Total t[5];

Faculty teacher1;

Administration admin1;

admin1.addstudentdetail(t[0], 25, "mukul roy");

teacher1.addresult(t[0], 10, 11, 12, 13, 19, 20, 21, 22, 8, 9, 10, 11, 7, 8, 7, 8, 12, 13, 12, 13, 91);

admin1.addstudentdetail(t[1], 26, "mukul kumar");

teacher1.addresult(t[1], 10, 11, 12, 13, 19, 20, 21, 22, 28, 29, 30, 31, 7, 8, 7, 8, 12, 13, 12, 13, 91);

admin1.addstudentdetail(t[2], 27, "swapnil");

teacher1.addresult(t[2], 10, 11, 12, 13, 19, 20, 21, 22, 8, 9, 10, 11, 7, 8, 7, 8, 12, 13, 12, 13, 91);

admin1.addstudentdetail(t[3], 28, "sonali");

teacher1.addresult(t[3], 10, 11, 12, 13, 19, 20, 21, 22, 28, 29, 30, 31, 7, 8, 7, 8, 12, 13, 12, 13, 91);

admin1.addstudentdetail(t[4], 29, "naman");

teacher1.addresult(t[4], 10, 11, 12, 1, 1, 0, 21, 22, 28, 29, 30, 31, 7, 8, 7, 8, 12, 13, 12, 13, 91);

for (int i = 0; i < 5; i++)

{

t[i].student::display();

}

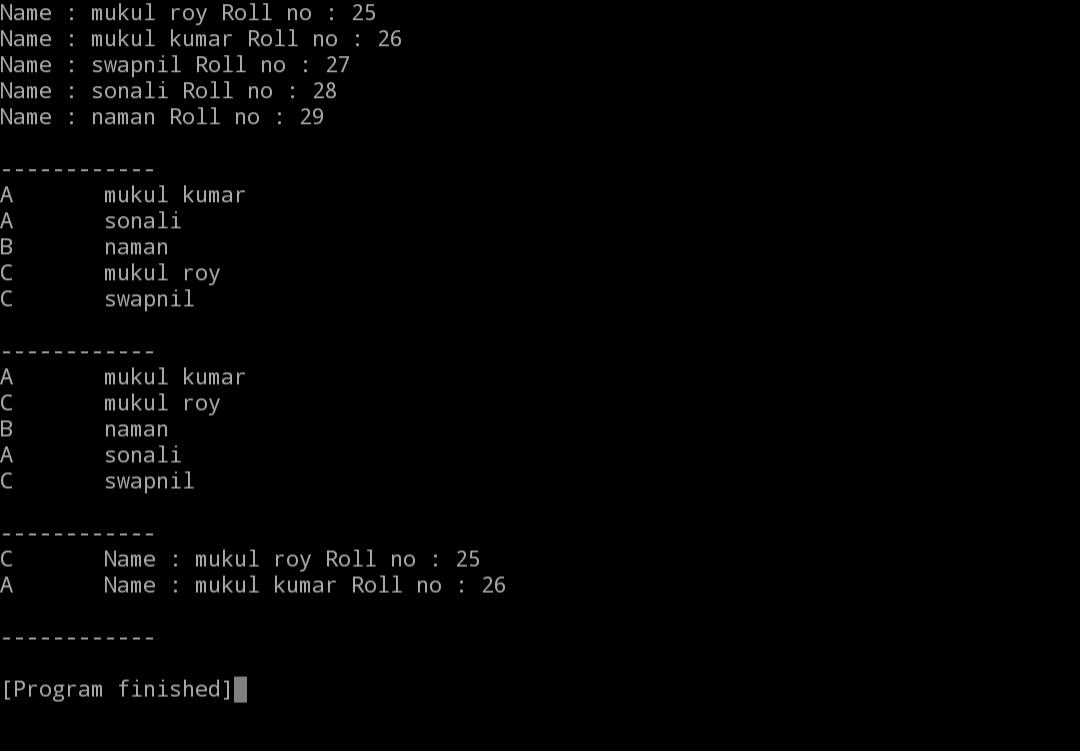
studgrade(t, 5);

studgrade\_lex(t, 5);

search(t, "mukul", 5);

return 0;

}



**Question 6** : Consider a case of single inheritance where Landline phone is a base class and Mobile phone is the derived class. Both the classes are as follow:

(a) Landline: It has subscriber name and number as data members. The member functions are to provide the features of calling on a subscriber's number and receiving a call.

Void call (int sub\_number);

Void receive();

(b) Mobile: Apart from inheriting the features of a Landline phone, it provides following additional features:

(i) Maintaining a phonebook to save the name and phone number of friends and relatives. For this, a data member of type array of strings has to be added.

(ii) Calling to a subscriber with its name.

Void call (string sub\_name);

This function first searches the name of the subscriber to be called in the phonebook to find the corresponding phone number and then, invokes the function “void call (int sub\_number)” by passing the searched phone number as argument.

(iii) Maintaining a list of last 20 dialled numbers. For this, a data member of type array of 20 integers has to be added. An entry will be made to this array each time whenever call() function will be invoked. In case of 21th entry to the array, the earliest entry will be replaced with the latest entry.

(iv) Calling on a number from the list of dialled numbers. This function first displays the list of dialled numbers and provides an option to choose a phone number from the list to which a call has to be made. Finally, it invokes call() function and passes the chosen phone number as an argument.

Finally, write the main program to show the features of each class.

**Code :**

#include <iostream>

using std::cin;

using std::cout;

using std::endl;

using std::string;

typedef struct phonebook

{

string name;

long long int number;

} phonebook;

typedef struct record

{

string type;

long long int number;

} record;

class landline

{

private:

record \*records\_ptr;

int record\_size = 0;

int flag = 0;

protected:

string name;

long long int number;

public:

void sub\_data(string name, long long int number)

{

this->name = name;

this->number = number;

}

void call(long long int sub\_number)

{

cout << "Dailing...." << endl;

if (record\_size == 0)

{

record\_size++;

records\_ptr = (record \*)malloc(record\_size \* sizeof(record));

records\_ptr[record\_size - 1].number = sub\_number;

records\_ptr[record\_size - 1].type = "incoming";

}

else if (record\_size <= 20 && flag == 0)

{

record\_size++;

records\_ptr = (record \*)realloc(records\_ptr, record\_size \* sizeof(record));

records\_ptr[record\_size - 1].number = sub\_number;

records\_ptr[record\_size - 1].type = "incoming";

if (record\_size == 20)

{

flag = 1;

}

}

else

{

record \*temp\_records\_ptr;

temp\_records\_ptr = (record \*)malloc(record\_size \* sizeof(record));

for (int i = 0; i < record\_size - 1; i++)

{

temp\_records\_ptr[i].number = records\_ptr[i + 1].number;

temp\_records\_ptr[i].type = records\_ptr[i + 1].type;

}

temp\_records\_ptr[record\_size - 1].number = sub\_number;

temp\_records\_ptr[record\_size - 1].type = "incoming";

records\_ptr = temp\_records\_ptr;

}

}

void recieve(long long int sub\_number)

{

if (record\_size == 0)

{

record\_size++;

records\_ptr = (record \*)malloc(record\_size \* sizeof(record));

records\_ptr[record\_size - 1].number = sub\_number;

records\_ptr[record\_size - 1].type = "recieving";

}

else if (record\_size <= 20 && flag == 0)

{

record\_size++;

records\_ptr = (record \*)realloc(records\_ptr, record\_size \* sizeof(record));

records\_ptr[record\_size - 1].number = sub\_number;

records\_ptr[record\_size - 1].type = "recieving";

if (record\_size == 20)

{

flag = 1;

}

}

else

{

record \*temp\_records\_ptr;

temp\_records\_ptr = (record \*)malloc(record\_size \* sizeof(record));

for (int i = 0; i < record\_size - 1; i++)

{

temp\_records\_ptr[i].number = records\_ptr[i + 1].number;

temp\_records\_ptr[i].type = records\_ptr[i + 1].type;

}

temp\_records\_ptr[record\_size - 1].number = sub\_number;

temp\_records\_ptr[record\_size - 1].type = "recieving";

records\_ptr = temp\_records\_ptr;

}

}

void callbyhistory()

{

int index;

for (int i = 0; i < record\_size; i++)

{

cout << i << "\t" << records\_ptr[i].number << endl;

}

cin >> index;

call(records\_ptr[index].number);

}

};

class mobile : public landline

{

private:

int phone\_book\_size = 0;

protected:

phonebook \*ptr = NULL;

public:

void contact ()

{

for (int i = 0;i<phone\_book\_size;i++)

{

cout<<ptr[i].name<<"\t"<<ptr[i].number <<endl;

}

}

void addcontact(string pname, long long int pno)

{

if (phone\_book\_size == 0)

{

++phone\_book\_size;

ptr = (phonebook \*)malloc(phone\_book\_size \* sizeof(phonebook));

}

else

{

++phone\_book\_size;

ptr = (phonebook \*)realloc(ptr, phone\_book\_size \* sizeof(phonebook));

}

if (ptr != NULL)

{

ptr[phone\_book\_size - 1].name = pname;

ptr[phone\_book\_size - 1].number = pno;

}

else

{

addcontact(pname, pno);

}

}

void call(string sub\_name)

{

for (int i = 0; i < phone\_book\_size; i++)

{

if (ptr[i].name == sub\_name)

{

landline::call(ptr[i].number);

return;

}

}

cout << "No Contact named " << sub\_name << endl;

return;

}

};

int main()

{

mobile m;

m.sub\_data("mukul",7493895160) ;

m.addcontact("mukul",7493895160) ;

m.addcontact("mukul kumar",7493895161) ;

m.addcontact("mukul roy",7493895162) ;

m.addcontact("mukul rajput",7493895163) ;

m.contact();

m.call("mukul");

m.call("mukul roy");

m.landline::call(8743287423);

m.recieve(2938420344);

m.landline::call(8756840345);

m.landline::call(8743287423);

m.landline::call(8743287423);

m.landline::call(8743287423);

m.landline::call(8743287423);

m.landline::call(8743287423);

m.landline::call(8743287423);

m.landline::call(8743287423);

m.landline::call(8743287423);

m.call("mukul rajput");

m.landline::call(8743287423);

m.call("mukul");

m.call("mukul roy");

m.landline::call(8743287423);

m.recieve(2938420344);

m.landline::call(8756840345);

m.call("mukul");

m.landline::call(8743287423);

m.callbyhistory();

m.callbyhistory();

return 0;

}

