|  |
| --- |
| **Assignment # 9**  ***Session*: Spring 2022 *Total marks*: 100**  ***Name*** : ***\_\_nimra maqbool\_ Roll number* : \_bsce21012\_\_** |

***Submission:***

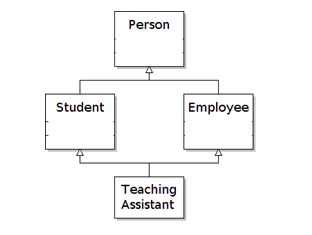
• *Email instructor or TA if there are any questions. You cannot look at others’ solutions or use others’ solutions, however, you can discuss it with each other. Plagiarism will be dealt with according to the course policy.*

*• 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 BSCE21001\_Assignment9.pdf.**

**TASK:**

Write a C++ program that implements the given class hierarchy.



A Teaching Assistant is an employee and a student. Both employee and student are persons.

Person is an abstract class. Person has a name, age, gender, education, language he speaks, etc. Since Person is an abstract class so make virtual functions.

Students have gpa, semester, field, etc.

Employees have salary, rank( like HOD or dean or assistant professor) etc.

Following methods should be implemented:

- Copy constructor

- Deep copy constructor

- Dectructor

- Function overriding

- Initializer List

- Deleting pointers

- Display() and setter/getter

- Make a menu driven code.

**Note:** Do not use strings. User char pointers instead.

|  |
| --- |
| **Function.h:**  #include <iostream>  using namespace std;  class person { protected:  char \*name;  int age;  char \*gender;  char \*education; //declaring  char \*language;  int size; public:  person() {  age = 0;  name = NULL;  gender = NULL; //making it null  education = NULL;  language = NULL;  size = 0;  }   person(int age1) : age(age1) {  name = new char[100];  gender = new char[100];  education = new char[100]; //ellocating memory  language = new char[100];  }   person(person &p1) {  age = p1.age;  size = p1.size;  name = new char[100];  for (int i = 0; i < '\0'; i++) {  name[i] = p1.name[i];  }  gender = new char[100];  for (int i = 0; i < '\0'; i++) {  gender[i] = p1.gender[i];  }  education = new char[100];  for (int i = 0; i < '\0'; i++) {  education[i] = p1.education[i];  }  language = new char[100];  for (int i = 0; i < '\0'; i++) {  language[i] = p1.language[i];  }   }   void setAge(int age2) {  cout << "ENTER AGE = ";  cin >> age2;  age = age2;  }   int getAge() {  return age;  }   void setGender(char \*&gender1) {  gender1 = new char[100];  cout<<"ENTER GENDER = ";  cin.ignore();  cin.getline(gender1,100);  for (int i = 0; i < 100; i++) {  gender[i] = gender1[i];  }   }   char \*getGender() {  return gender;  }   void setName(char \*&name1) {  name1 = new char[100];  cout<<"ENTER NAME = ";  cin.getline(name1,100);  for (int i = 0; i < 100; i++) {  name[i] = name1[i];  }  delete[] name1;  }   char \*getName() {  return name;  }   void setEducation(char \*&education1) {  education1 = new char[100];  cout<<"ENTER EDUCATION = ";  cin.getline(education1,100);  for (int i = 0; i < 100; i++) {  education[i] = education1[i];  }  delete[] education1;   }   char \*getEducation() {  return education;  }   void setLanguage(char \*&language1) {  language1 = new char[100];  cout<<"ENTER LANGUAGE = ";  cin.getline(language1,100);  for (int i = 0; i < 100; i++) {  language[i] = language1[i];  }   delete[] language1;  }   char \*getLanguage() {  return language;  }   virtual void display() {  cout << "AGE = " << age << endl;  int size1;  size1=sizeof(name);  cout << "NAME = ";  for (int i = 0; i < size1; i++) {  cout << name[i];  }  cout<<" ";  int size2;  size2=sizeof(gender);  cout << "GENDER = ";  for (int i = 0; i < size2; i++) {  cout << gender[i];  }  cout<<" ";  int size3;  size3=sizeof(education);  cout << "EDUCATION = ";  for (int i = 0; i < size3-1; i++) {  cout << education[i];  }  cout<<" ";  int size;  size=sizeof (language);  cout << "LANGUAGE = ";  for (int i = 0; i < size; i++) {  cout << language[i];  }  cout<<endl;  }   ~person() {  delete[]name;  delete[]education;  delete[]gender;  delete[]language;  } };  class student:virtual public person{ protected:  float gpa;  int semester;  char \*field; public:  student(){  gpa=0;  semester=0;  field=NULL;  }  student(int sem,float Gpa):semester(sem),gpa(Gpa){  field=new char[100];  }  student(student &s1){  gpa=s1.gpa;  semester=s1.semester;  field = new char[100];  for (int i = 0; i < '\0'; i++) {  field[i] = s1.field[i];  }  }  void setSemester(int sem){  cout<<"ENTER SEMESTER = ";  cin>>sem;  semester=sem;  }  void setGpa(float Gpa){  cout<<"ENTER GPA = ";  cin>>Gpa;  gpa=Gpa;  }  void setField(char \*f){  f=new char[100];  cout << "ENTER FIELD = ";  cin.ignore();  cin.getline(f,100);  for(int i=0;i<100;i++){  field[i]=f[i];  }  delete [] f;  }  int getSemester(){  return semester;  }  float getGpa(){  return gpa;  }  char\* getField(){  return field;  }  void display(){  cout<<"SEMESTER = "<<semester<<" ";  cout<<"GPA = "<<gpa<<" ";  int size;  cout<<"FIELD = ";  size= sizeof(field);  for(int i=0;i<size;i++){  cout<<field[i];  }  cout<<endl;  }  ~student(){  delete [] field;  }  }; class employee:virtual public person { protected:  int salary;  char\* rank; public:  employee(){  salary=0;  rank=NULL;  }  employee(int s):salary(s){  rank= new char[100];  }  employee(employee &E){  salary=E.salary;  rank = new char[100];  for (int i = 0; i < '\0'; i++) {  rank[i] = E.rank[i];  }  }  void setSalary(int s){  cout<<"ENTER SALARY = ";  cin>>s;  salary=s;  }  int getSalary(){  return salary;  }  void setRank(char \*R){  R=new char[100];  cout << "ENTER RANK = ";  cin.ignore();  cin.getline(R,100);  for(int i=0;i<100;i++){  rank[i]=R[i];  }  delete [] R;  }   char\* getRank(){  return rank;  }  void display(){  cout<<"SALARY = "<<salary<<" ";  int size;  size= sizeof(rank);  cout<<"RANK = ";  for(int i=0;i<size;i++){  cout<<rank[i];  }  cout<<endl;  }  ~employee(){  delete [] rank;  } }; class teachingAssistant: public student, public employee { public:  teachingAssistant(int sem,float Gpa,int s): student(sem,Gpa), employee(s){  }  void display(){  cout<<"\nSALARY = "<<salary<<" ";  int size;  size= sizeof(rank);  for(int i=0;i<size;i++){  cout<<rank[i];  }  int size1;  size1= sizeof(field);  for(int i=0;i<size1;i++){  cout<<field[i];  }  cout<<endl;  cout<<"SEMESTER = "<<semester<<" ";  cout<<"GPA = "<<gpa<<" "<<endl;  }  };   * **In function.h, I have made 4 classes 1 class is parent of student and employee and the next is inheriting both student and employee.** * **In main I have made a person class having name, age, gender, language, and education as attributes.** * **Then I have made a default parameter and set the values to zero and pointers to null.** * **Then I have made parametrized constructor in which I have allocated memory then in copy constructor I have copied the array in the other array.** * **Then I have made getter setters and made a virtual function a display.** * **Then I have made another class of student inheriting person class, in which I have gpa, semester, and field.** * **Then I have made again a default and parametrized and copy constructor** * **Then made a display function and same thin goes for employee class** * **And in last class I have inherited the student and employee class** * **And then display the function**   **Main.cpp:**  #include <iostream> #include “Functions.h”  using namespace std;  int main() {  int age = 23;  int age2;  char \*gender1;  char \*name1;  char \*education1;  char \*language1;  person p;  person p1(23);  student s2;  student s1(2, 2.93);  student s3 = s1;  float Gpa;  int sem;  char \*f;  int s;  char \*R;  employee e;  employee E(20000);  teachingAssistant T(2, 2.0, 2345);  int opt;  do {  cout << “\nFROM WHICH CONSTRUCTOR DO YOU WANT TO CALL THE FUNCTIONS?” << endl;  cout << “1.CONSTRUCTOR..” << endl;  cout << “2.DEEP COPY CONSTRUCTOR..” << endl;  cout << “3.EXIT..” << endl;  cin >> opt;  switch (opt) {  case 1: {  p1.setAge(age2);  p1.getAge();  p1.setGender(gender1);  p1.getGender();  p1.setName(name1);  p1.getName();  p1.setEducation(education1);  p1.getEducation();  p1.setLanguage(language1);  p1.getLanguage();  p1.display();   s1.setGpa(Gpa);  s1.getGpa();  s1.setSemester(sem);  s1.getSemester();  s1.setField(f);  s1.getField();  s1.display();   E.setSalary(s);  E.getSalary();  E.setRank(f);  E.getRank();  E.display();   T.display();   break;  }  case 2: {  person p2 = p1;  p2.setAge(age2);  p2.getAge();  p2.setGender(gender1);  p2.getGender();  p2.setName(name1);  p2.getName();  p2.setEducation(education1);  p2.getEducation();  p2.setLanguage(language1);  p2.getLanguage();  p2.display();   student s3 = s1;  s3.setGpa(Gpa);  s3.getGpa();  s3.setSemester(sem);  s3.getSemester();  s3.setField(f);  s3.getField();  s3.display();   employee E1 = E;  E1.setSalary(s);  E1.getSalary();  E1.setRank(f);  E1.getRank();  E1.display();    break;  }  case 3: {  cout<<”YOU CHOOSE TO EXIT ..”<<endl;  exit(3);  break;  }  default: {  cout<<”you have entered invalid arguments..”<<endl;  break;  }  }  }while(opt>=1 && opt<=3);  return 0; }   * **In main I have made the menus and called the functions.**   **Output:**  **Text  Description automatically generated** |