|  |
| --- |
| **Assignment # 5**  ***Session*: Spring 2022 *Total marks*: 100**  ***Name*** : ***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Roll number* : \_\_\_\_\_\_\_\_\_\_\_\_\_** |

***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\_Assignment5.pdf.**

**TASK A**

Create a Duck class.

The attributes of Duck are: number and energy

Implement the following methods and make UML diagram against each added function:

1- Constructor to assign a number to duck and set its energy to 10. Number of duck will be unique. (e.g You can increment the number of duck each time to be used)

2- eating //when duck eats, energy increases by 3

3- swimming //when duck swims, energy decreases by 2

4- walking //when duck walks, energy decreases by 1

5- print //prints energy of duck

6- getEnergy // getter function for energy

7- Destructor

|  |
| --- |
| **Function.cpp:**  duck::duck(int eng,int num){  char ch; eng=10;  energy=eng;  num=1;  number=num;  do {  cout << "\nDo you want to use another duck? " << "\nEnter y for yes and n for no " << endl;  cin >> ch;   if (ch =='y') {  number++;  cout << "NUMBER OF THE DUCK = " << number << endl;  cout << "THE ENERGY OF THE DUCK = " << energy << endl;  } else {  cout << "YOU DON'T WANTED THE OTHER DUCK" << "\nTHE NUMBER OF DUCK IS = " << number << endl;  cout << "THE ENERGY OF THE DUCK = " << energy << endl; break; }  }  while(ch=='y' ); }  void duck::eating(){  char ch;  cout<<"\nDO YOU WANT DUCK TO EAT ?"<<endl;  cout<<"ENTER Y FOR YES AND N FOR NO "<<endl;  cin>>ch;  if(ch=='y'){  energy=energy+3;  cout<<"THE ENERGY OF THE DUCK NUMBER "<<number<<" AFTER EATING = "<<energy<<endl;  }  else{  cout<<"THE ENERGY OF THE DUCK = "<<energy<<endl;  } } void duck::swimming(){  char ch;  cout<<"\nDO YOU WANT DUCK TO SWIM ?"<<endl;  cout<<"ENTER Y FOR YES AND N FOR NO "<<endl;  cin>>ch;  if(ch=='y'){  energy=energy-2;  cout<<"THE ENERGY OF THE DUCK NUMBER "<<number<<" AFTER SWIMMING = "<<energy<<endl;  }  else{  cout<<"THE ENERGY OF THE DUCK = "<<energy<<endl;  } } void duck::walking(){  char ch;  cout<<"\nDO YOU WANT DUCK TO WALK ?"<<endl;  cout<<"ENTER Y FOR YES AND N FOR NO "<<endl;  cin>>ch;  if(ch=='y'){  energy=energy-1;  cout<<"THE ENERGY OF THE DUCK NUMBER "<<number<<" AFTER WALKING = "<<energy<<endl;   }  else{  cout<<"THE ENERGY OF THE DUCK = "<<energy<<endl;  } }  void duck::print(){  cout<<"\nTHE ENERGY OF THE DUCK NUMBER "<<number<<" AFTER EATING = "<<energy<<endl;  } int duck::getEnergy(){  return energy; } duck::~duck(){  cout<<"THE DESTRUCTOR IS CALLED "<<endl; }   * **I have make a constructor and given it path to the class.** * **Then I set the energy to 10 and then I set the number and ask the user that If he want to use the duck if yes then incremented the number else don’t change it.** * **Then I have make an eating function and ask the user if he want the duck to eat food if yes then incremented the energy by 3,else remained same.** * **Then made an swimming function and ask the user if he wants the duck to swim if yes then decremented the energy by 2 and else remains same.** * **Then I make an walking function and ask the user that if he wants the duck to walk if yes then decremented the energy by 1 else remains same.** * **Then I make a print functions and printed the energy of duck.** * **Then an get energy function to return the energy** * **Then an destructor displaying that destructor is called.**   **function.h:**  class duck { private:  int energy;  int number; public:  duck(int eng,int num);  void eating ();  void swimming();  void walking();  void print();  int getEnergy();  ~duck(); };   * **I have made an duck class having 2 private members of number and energy.** * **Then in public I have declared all the functions including the constructor and destructors.**   **main.cpp:**  int eng; int num; duck d ( eng, num);  d.eating();  d.swimming();  d.walking();  d.print(); d.getEnergy(); d.~duck();   * **I have initialized the num and eng.** * **Then called all the functions including constructors and destructors.**   **output:**  **Text  Description automatically generated**  **Uml diagram:**  **A picture containing table  Description automatically generated** |

### **TASK B**

### Add Copy Constructor in your duck class to assign values of attributes of one object to another.

Also, modify functions of Task A in such a way that functions must be Inline member functions (wherever possible).

|  |
| --- |
| **Function.h:**  using namespace std; class duck { private:  int energy;  int number; public:  duck(int eng,int num);  duck(duck &d);  void eating ();  void swimming();  void walking();  void print();  int getEnergy();  ~duck(); }; inline duck::duck(int eng,int num){  char ch;  eng=10;  energy=eng;  num=1;  number=num;  do {  cout << "\nDo you want to use another duck? " << "\nEnter y for yes and n for no " << endl;  cin >> ch;   if (ch =='y') {  number++;  cout << "NUMBER OF THE DUCK = " << number << endl;  cout << "THE ENERGY OF THE DUCK = " << energy << endl;  } else {  cout << "YOU DON'T WANTED THE OTHER DUCK" << "\nTHE NUMBER OF DUCK IS = " << number << endl;  cout << "THE ENERGY OF THE DUCK = " << energy << endl;  break; }  }  while(ch=='y' ); } inline duck::duck(duck &d){  cout<<"\nCOPY CONSTRUCTED IS CALLED"<<endl;  energy =d.energy;  number =d.number; } inline void duck::eating(){  char ch;  cout<<"\nDO YOU WANT DUCK TO EAT ?"<<endl;  cout<<"ENTER Y FOR YES AND N FOR NO "<<endl;  cin>>ch;  if(ch=='y'){  energy=energy+3;  cout<<"THE ENERGY OF THE DUCK NUMBER "<<number<<" AFTER EATING = "<<energy<<endl;  }  else{  cout<<"THE ENERGY OF THE DUCK = "<<energy<<endl;  } } inline void duck::swimming(){  char ch;  cout<<"\nDO YOU WANT DUCK TO SWIM ?"<<endl;  cout<<"ENTER Y FOR YES AND N FOR NO "<<endl;  cin>>ch;  if(ch=='y'){  energy=energy-2;  cout<<"THE ENERGY OF THE DUCK NUMBER "<<number<<" AFTER SWIMMING = "<<energy<<endl;  }  else{  cout<<"THE ENERGY OF THE DUCK = "<<energy<<endl;  } } inline void duck::walking(){  char ch;  cout<<"\nDO YOU WANT DUCK TO WALK ?"<<endl;  cout<<"ENTER Y FOR YES AND N FOR NO "<<endl;  cin>>ch;  if(ch=='y'){  energy=energy-1;  cout<<"THE ENERGY OF THE DUCK NUMBER "<<number<<" AFTER WALKING = "<<energy<<endl;   }  else{  cout<<"THE ENERGY OF THE DUCK = "<<energy<<endl;  } }  inline void duck::print(){  cout<<"\nTHE ENERGY OF THE DUCK NUMBER "<<number<<" AFTER EATING = "<<energy<<endl; } inline int duck::getEnergy(){  return energy; } inline duck::~duck(){  cout<<"THE DESTRUCTOR IS CALLED "<<endl; }   * **I have made an duck class having 2 private members of number and energy.** * **Then in public I have declared all the functions including the constructor and destructors.** * **I have make an inline constructor and given it path to the class.** * **Then I set the energy to 10 and then I set the number and ask the user that If he want to use the duck if yes then incremented the number else don’t change it.** * **Then I have make an inline eating function and ask the user if he want the duck to eat food if yes then incremented the energy by 3,else remained same.** * **Then made an inline swimming function and ask the user if he wants the duck to swim if yes then decremented the energy by 2 and else remains same.** * **Then I make an inline walking function and ask the user that if he wants the duck to walk if yes then decremented the energy by 1 else remains same.** * **Then I make an inline print functions and printed the energy of duck.** * **Then an inline get energy function to return the energy** * **Then an inline destructor displaying that destructor is called.**   **main.cpp:**  int main() {    int eng;  int num;  duck d ( eng, num);  duck d1 =d;  d1.eating();  d1.swimming();  d1.walking();  d1.print();  d1.getEnergy();  d1.~duck();   return 0; }   * **I have initialized the num and eng.** * **Then called all the functions including constructors and destructors.**   **output:**  **Text  Description automatically generated** |

|  |
| --- |
|  |