**LAB 2: WRITE OOP PROGRAMS WITH C++ AND JAVA**

**PRACTICES WITH SCOPE RESOLUTION OPERATOR, DESTRUCTOR, STATIC MEMBER, REFERENCE**

1. **Declare the class MyUtil as a utility class that gives services using several static methods. The class should have the following static methods:**
2. **factorial, that receives one int value and returns its factorial. The returned value type is long.**
3. **isPrime, that receives a number and check whether it is a prime number. The returned value type is boolean.**
4. **isTriangle, that receives three numbers and check whether these numbers can create a triangle. The returned value type is boolean.**

**You should check these methods within an application, which you should create as a separate class.**

|  |  |
| --- | --- |
| **//JAVA: Static member**  **class** MyUtil  {  **static** **long** factorial(**int** num)  {  **long** result=1;  **for**(**int** i=2;i<=num;i++) result\*=i;  **return** result;  }  **static** **boolean** isPrime(**long** number)  {  **long** numberSqrt = (**long**)Math.*sqrt*(number);  **boolean** result = **true**;  **for**(**int** i=2; i<numberSqrt && result; i++)  **if**(number%i==0) result = **false**;  **return** result;  }  **static** **boolean** isTriangle(**double** num1, **double** num2, **double** num3)  {  **return** (num1>(num2+num3) && num2>(num1+num3) && num3>(num2+num1));  }  }  **public** **class** MyUtilDemo  {  **public** **static** **void** main(String args[])  {  **double** a=12,b=2,c=8;  System.*out*.println("12, 2 and 8 create a triangle is : " + MyUtil.*isTriangle*(12,2,8));  System.*out*.println("The factorial of 6 is : " + MyUtil.*factorial*(6));  System.*out*.println("Saying that 17 is a prime number is : " + MyUtil.*isPrime*(17));  }  } | **//C++**  **// DO IT BY YOURSELF** |

1. **Re-write the below program in C++ that all member functions are defined outside the class (using the scope resolution operator)**

**//C++**

#include <iostream>

#include <conio.h>

using namespace std;

class Rectangle

{

private:

double width, height;

public:

Rectangle (double x, double y)

{

width =x;

height =y;

}

double perimeter()

{

return 2\*(width + height);

}

double area()

{

return width \* height;

}

};

main()

{

Rectangle rec1(23,20);

Rectangle rec2 (40,50);

cout<<"area of rec1 is : "<< rec1.area();

cout<<"\n area of rec2 is : "<<rec2.area();

getch();

}

|  |  |
| --- | --- |
| **//C++**  #include <iostream>  #include <conio.h>  using namespace std;  class Rectangle  {  private:  double width, height;  public:  Rectangle (double x, double y)  {  width =x;  height =y;  }  double perimeter()  {  return 2\*(width + height);  }  double area()  {  return width \* height;  }  };  main()  {  Rectangle rec1(23,20);  Rectangle rec2 (40,50);  cout<<"area of rec1 is : "<< rec1.area();  cout<<"\n area of rec2 is : "<<rec2.area();  getch();  } | **//C++: Scope resolution operator**  **//DO IT BY YOURSELF**  .....  ......  .......  class Rectangle  {  private:  double width, height;  public:  Rectangle (double x, double y) ;  double perimeter();  double area();  };  main()  {  .......  .......  }  .......  .......  double Rectangle::perimeter()  {  return 2\*(width + height);  }  double Rectangle::area()  {  return width \* height;  } |

**3. Linked List Data Structure. Creation of Linked List Using C++. Write methods for the creation, display, insertion, deletion of the linked list.**

|  |  |
| --- | --- |
| **//C++**  #include <stdio.h>  #include <conio.h>  #include <iostream>  using namespace std;  struct node  {  int data;  node \*next;  };    class List  {  private:  node \*head, \*tail;  public:  List()  {  head=NULL;  tail=NULL;  }  ~List()  {  delete head;  delete tail;  }    //creation of a new node for the linked list  void createnode(int value)  {  node \*temp=new node;  temp->data=value;  temp->next=NULL;    if(head==NULL)  {  head=temp;  tail=temp;  temp=NULL;  }  else  {  tail->next=temp;  tail=temp;  }  }  //displaying nodes of linked list  void display()  {  node \*temp=new node;  temp=head;  while(temp!=NULL)  {  cout<<temp->data<<"-->";  temp=temp->next;  }  cout<<"NULL\n\n";  }    //insert at the start  void insert\_start(int value)  {  node \*temp=new node;  temp->data=value;  temp->next=head;  head=temp;  }    //insert at the end  void insert\_end(int value)  {  node \*temp=new node;  temp->data=value;  temp->next=NULL;    tail->next=temp;  tail=temp;  }    //insert at the particular position  void insert\_position(int pos, int value)  {  node \*pre=new node;  node \*cur=new node;  node \*temp=new node;  cur=head;  for(int i=1;i<pos;i++)  {  pre=cur;  cur=cur->next;  }  temp->data=value;  pre->next=temp;  temp->next=cur;  }  //delete at the start  void delete\_first()  {  node \*temp=new node;  temp=head;  head=head->next;  delete temp;  }    //delete at the end  void delete\_last()  {  node \*current=new node;  node \*previous=new node;  current=head;  while(current->next!=NULL)  {  previous=current;  current=current->next;  }  tail=previous;  previous->next=NULL;  delete current;  }    //Deletion at a particular position    void delete\_position(int pos)  {  node \*current=new node;  node \*previous=new node;  current=head;  for(int i=1;i<pos;i++)  {  previous=current;  current=current->next;  }  previous->next=current->next;  }    };      int main()  {  List L;  int n;  cout<<"List size = "; cin>>n;  //input  for (int i= 1;i<=n; i++)  {  int value;  cout<<"\nValue of node " + i; cin>>value;    L.createnode(value);  }  //display  L.display();    L.insert\_start(8);  L.insert\_end(5);  L.insert\_position(3,6);  L.display();  L.delete\_first();  L.delete\_last();  L.delete\_position(3);  L.display();  getch();  return 0;  } | **//JAVA**  **//COMPLETE IT BY YOURSELF**  **import** java.util.Scanner;  **public** **class** LinkedList  {  **private** Node head, tail;  **static** **class** Node {  **int** data;  Node next;  Node(**int** d) { data = d; next=**null**; } *// Constructor*  }    **public** LinkedList()  {  head=**null**;  tail=**null**;  }    *//creation of a new node for the linked list*  **public** **void** createnode(**int** value)  {  Node temp=**new** Node(value);    **if**(head==**null**)  {  head=temp;  tail=temp;  temp=**null**;  }  **else**  {  tail.next=temp;  tail=temp;  }  }  *//displaying nodes of linked list*  **void** display()  {  Node temp=head;  **while**(temp!=**null**)  {  System.*out*.print(temp.data + "-->");  temp=temp.next;  }  System.*out*.println("NULL");  }    */\**  *//insert at the start*  *void insert\_start(int value)*  *{*  *...*  *}*    *//insert at the end*  *void insert\_end(int value)*  *{*  *...*  *}*    *//insert at the particular position*  *void insert\_position(int pos, int value)*  *{*  *...*  *}*  *//delete at the start*  *void delete\_first()*  *{*  *...*  *}*    *//delete at the end*  *void delete\_last()*  *{*  *...*  *}*    *//Deletion at a particular position*    *void delete\_position(int pos)*  *{*  *...*  *}*    *\*/*    **public** **static** **void** main(String[] arg)  {  LinkedList L = **new** LinkedList() ;  **int** n;  Scanner keyboard = **new** Scanner(System.*in*);  System.*out*.print("n = ");  n = keyboard.nextInt();  *//input*  **for** (**int** i= 1;i<=n; i++)  {  **int** value;  System.*out*.print("value = ");  value = keyboard.nextInt();    L.createnode(value);  }  *//display*  L.display();    *L.insert\_start(8);*  *L.insert\_end(5);*  *L.insert\_position(3,6);*  *L.display();*  *L.delete\_first();*  *L.delete\_last();*  *L.delete\_position(3);*  *L.display();*    }  } |

**4.   Create a Stack using C++ and Java with the following operations:**

**empty() – Returns whether the stack is empty**

**size() – Returns the size of the stack**

**top() – Returns a reference to the top most element of the stack**

**push(g) – Adds the element ‘g’ at the top of the stack**

**pop() – Deletes the top most element of the stack**