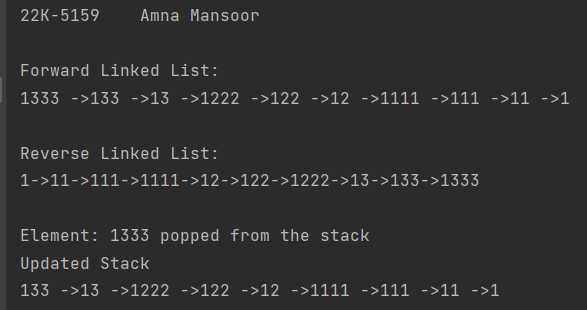
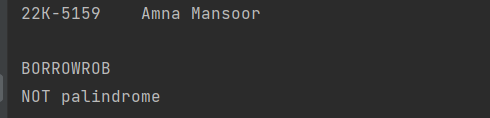
***Question 1: (from page: 3)***

import java.util.Stack;  
  
public class q1 {  
 public static void main(String[] args) {  
 System.*out*.println("22K-5159 Amna Mansoor\n");  
 StackLL obj=new StackLL();  
 //Insert 10 Integers values in the stack  
 obj.push(1);  
 obj.push(11);  
 obj.push(111);  
 obj.push(1111);  
 obj.push(12);  
 obj.push(122);  
 obj.push(1222);  
 obj.push(13);  
 obj.push(133);  
 obj.push(1333);  
  
 //Forward Linked List  
 System.*out*.println("Forward Linked List:");  
 obj.Display();  
  
 // Display the stack in reverse direction  
 System.*out*.println("\n\nReverse Linked List:");  
 obj.Reverse();  
  
 //Pop top element from the stack  
 int popped= obj.pop();  
 System.*out*.println("\n\nElement: "+popped+" popped from the stack");  
 System.*out*.println("Updated Stack");  
 obj.Display();  
  
 }  
}  
class StackLL{  
 class Node{  
 Node next;  
 int data;  
 Node link;  
 }  
 static Node *head*=null;  
 Node top;  
 StackLL(){  
 this.top=null;  
 }  
 public void push(int data){  
 Node temp=new Node();  
 if(temp==null){  
 System.*out*.println("Heap overflow");  
 return;  
 }  
 temp.data=data;  
 temp.link=top;  
 top=temp;  
  
 }  
 public int pop(){  
 if(top==null){  
 System.*out*.println("Stack Underflow");  
 return -1;  
 }  
 int poptop =top.data;  
 top=top.link;  
 return poptop;  
  
 }  
 void Reverse(){  
 if(top==null){  
 System.*out*.println("Stack Underflow");  
 return ;  
 }  
 Stack<Integer> stk=new Stack<>();  
 Node temp=top;  
 while (temp!=null){  
 stk.push(temp.data);  
 temp=temp.link;  
 }  
 while(!stk.isEmpty()){  
 System.*out*.print(stk.pop());  
 if (!stk.isEmpty()){  
 System.*out*.print("->");  
 }  
 }  
  
 }  
 public void Display(){  
 if(top==null){  
 System.*out*.println("Stack Underflow");  
 return;  
 }  
 Node temp=top;  
 while(temp!=null){  
 System.*out*.print(temp.data);  
 temp=temp.link;  
 if (temp!=null)  
 System.*out*.print(" ->");  
 }  
  
 }  
  
}



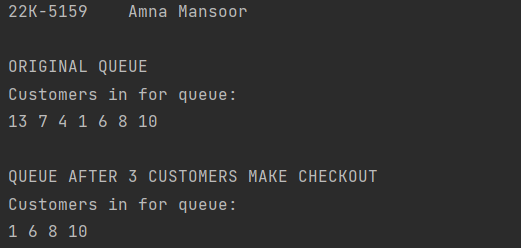
***Task 1:***

public class Task1 {  
 public static void main(String[] args){  
 System.*out*.println("22K-5159 Amna Mansoor\n");  
 char ip[]="BORROWROB".toCharArray();  
 System.*out*.println(ip);  
 if (Stackk.*isPalindrome*(ip)==1){  
 System.*out*.println("it is palindrome");  
 }  
 else{  
 System.*out*.println("NOT palindrome");  
 }  
  
 }  
}  
class Stackk{  
 static final int *MAX*=100;  
 static int *top*;  
 static char *a*[]=new char[*MAX*];  
  
 boolean isEmpty(){  
 return (*top*<0);  
 }  
 Stackk(){  
 *top*=-1;  
 }  
 static boolean push(char x){  
 if (*top*>=(*MAX*-1)){  
 System.*out*.println("Stack Overflow");  
 return false;  
 }  
 else {  
 *a*[++*top*]=x;  
// System.out.print(x+ "->");  
 return true;  
 }  
 }  
 static char pop(){  
 if (*top*<0){  
 System.*out*.println("Stack Underflow");  
 return 0;  
 }  
 else {  
 char x=*a*[*top*--];  
 return x;  
 }  
 }  
 static int isPalindrome(char ip[]){  
 int length=ip.length;  
  
  
 int i,mid=length/2;  
 for (i=0;i<mid;i++){  
 *push*(ip[i]);  
 }  
 if (length%2!=0){  
 i++;  
 }  
 while (i<length){  
 char ele=*pop*();  
 if (ele!=ip[i])  
 return 0;  
 i++;  
 }  
 return 1;  
 }  
  
  
}

****

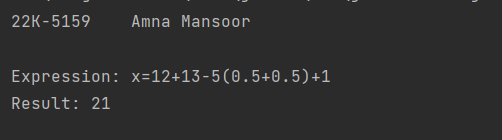
***Task 2:***

import java.util.Queue;  
  
public class Task2 {  
 public static void main(String[] args) {  
 System.*out*.println("22K-5159 Amna Mansoor\n");  
  
 QueueArray qa=new QueueArray(7);  
 qa.enqueue(13);  
 qa.enqueue(7);  
 qa.enqueue(4);  
 qa.enqueue(1);  
 qa.enqueue(6);  
 qa.enqueue(8);  
 qa.enqueue(10);  
  
 System.*out*.println("ORIGINAL QUEUE");  
 qa.Display();  
  
 qa.dequeue();  
 qa.dequeue();  
 qa.dequeue();  
  
 System.*out*.println("\nQUEUE AFTER 3 CUSTOMERS MAKE CHECKOUT");  
 qa.Display();  
 }  
}  
class QueueArray{  
 int front, rear, capacity;  
 int[] queue;  
  
 public QueueArray(int cap){  
 front=rear=0;  
 capacity=cap;  
 queue=new int[capacity];  
 }  
 public void enqueue(int data){  
 if (capacity==rear){  
 System.*out*.println("Queue is full");  
 return;  
 }  
 else{  
 queue[rear]=data;  
 rear++;  
 }  
 }  
 public void dequeue(){  
 if(front==rear){  
 System.*out*.println("Queue is Empty");  
 return;  
 }  
 else{  
 for (int i=0;i<rear-1;i++){  
 queue[i]=queue[i+1];  
 }  
 if(rear<capacity){  
 queue[rear]=0;  
 }  
 rear--;  
 }  
 }  
 public void Display(){  
 if(front==rear){  
 System.*out*.println("Queue is Empty");  
 return;  
 }  
 System.*out*.println("Customers in for queue: ");  
 for(int i=front;i<rear;i++){  
 System.*out*.print(queue[i]+" ");  
 }  
 System.*out*.println();  
 }  
}



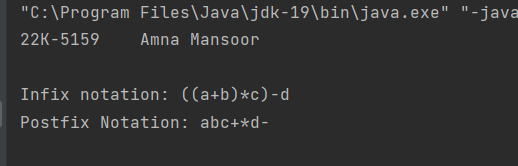
***Task 3:***

class Node{  
 char data;  
 Node next;  
  
 public Node(char data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
class Stack{  
 Node top;  
 void push(char data){  
 Node newNode=new Node(data);  
 newNode.next=top;  
 top=newNode;  
 }  
 char pop(){  
 if (isEmpty()){  
 System.*out*.println("Stack is Empty");  
 }  
 char data=top.data;  
 top=top.next;  
 return data;  
 }  
  
 private boolean isEmpty() {  
 return top==null;  
 }  
}  
public class Task3 {  
 public static void main(String[] args) {  
 System.*out*.println("22K-5159 Amna Mansoor\n");  
  
 String expression="x=12+13-5(0.5+0.5)+1";  
 Stack s=new Stack();  
 int result=0, num=0;  
 int sign=1; //sign variable for addition and subtraction  
 int oper=1; //operation variable for multiplication  
  
 for (char c:expression.toCharArray()){  
 if(Character.*isDigit*(c)){  
 num=num\*10+(c-'0');  
 }else {  
 result+=(sign\*num\*oper);  
 num=0;  
 if (c=='-'){  
 sign=-1;  
 } else if (c=='+') {  
 sign=1;  
 } else if (c=='(') {  
 s.push((char)oper);  
 oper=1;  
 }else if (c==')') {  
 int prev\_oper=s.pop();  
 oper=prev\_oper;  
 } else if (c=='\*') {  
 oper=1;  
 }  
 }  
 }  
 result+=(sign\*num\*oper);  
 System.*out*.println("Expression: "+expression);  
 System.*out*.println("Result: "+result);  
  
 }  
}



***Task 4:***

class Node{  
 char data;  
 Node next;  
  
 public Node(char data) {  
 this.data = data;  
 this.next=null;  
 }  
}  
class LinkedList{  
 Node head;  
 Node tail;  
  
 public LinkedList() {  
 head=null;  
 tail=null;  
 }  
 public void Add(char data){  
 Node newNode=new Node(data);  
 if (tail==null){  
 head=newNode;  
 tail=newNode;  
 }else{  
 tail.next=newNode;  
 tail=newNode;  
 }  
 }  
 public char Remove(){  
 if (head==null){  
 System.*out*.println("Queue is empty");  
 }  
 char data= head.data;  
 head=head.next;  
 if (head==null){  
 tail=null;  
 }  
 return data;  
 }  
 public boolean isEmpty(){  
 return head==null;  
 }  
  
}  
public class Task4 {  
 public static void main(String[] args) {  
 System.*out*.println("22K-5159 Amna Mansoor\n");  
 String notation="((a+b)\*c)-d";  
 System.*out*.println("Infix notation: "+notation);  
 String pE=*Conversion*(notation);  
 System.*out*.println("Postfix Notation: "+pE);  
 }  
 static int Precedence(char operator){  
 switch (operator){  
 case'+':  
 case'-':  
 return 1;  
 case'\*':  
 case'/':  
 return 2;  
 }  
 return 0;  
 }  
 static String Conversion(String infix){  
 LinkedList postfix=new LinkedList();  
 LinkedList operQueue=new LinkedList();  
  
 for (char c: infix.toCharArray()){  
 if (Character.*isLetter*(c)){  
 postfix.Add(c);  
 } else if (c=='(') {  
 operQueue.Add(c);  
 } else if (c==')') {  
 while (!operQueue.isEmpty()&&operQueue.head.data!='('){  
 postfix.Add(operQueue.Remove());  
 }  
 operQueue.Remove();  
 } else if (*Precedence*(c)>0) {  
 while (!operQueue.isEmpty()&&*Precedence*(c)<=*Precedence*(operQueue.head.data)){  
 postfix.Add(operQueue.Remove());  
 }  
 operQueue.Add(c);  
 }  
 }  
 while(!operQueue.isEmpty()){  
 postfix.Add(operQueue.Remove());  
 }  
 String result="";  
 Node current=postfix.head;  
 while (current!=null){  
 result+=current.data;  
 current=current.next;  
 }  
 return result;  
 }  
}



***Task 5:***

public class Task5 {  
 public static void main(String[] args) {  
 System.*out*.println("22K-5159 Amna Mansoor\n");  
 priorityQueue pQ=new priorityQueue();  
 pQ.enqueue(new Task(1,20,30,512,2));  
 pQ.enqueue(new Task(2,15,20,256,1));  
 pQ.enqueue(new Task(3,25,40,768,3));  
 pQ.enqueue(new Task(4,18,25,384,2));  
 System.*out*.println("Execution Tasks:");  
 while(!pQ.isEmpty()){  
 Task t=pQ.dequeue();  
 System.*out*.println(t);  
 }  
 }  
}  
class Task{  
 int TaskID, ExecutionTime, CPU\_Usage, Memory, Storage;  
  
 public Task(int TaskID, int ExecutionTime, int CPU\_Usage, int Memory, int Storage) {  
 this.TaskID = TaskID;  
 this.ExecutionTime = ExecutionTime;  
 this.CPU\_Usage = CPU\_Usage;  
 this.Memory = Memory;  
 this.Storage = Storage;  
 }  
  
 @Override  
 public String toString() {  
 return "|" +  
 "TaskID=" + TaskID +  
 " | ExecutionTime=" + ExecutionTime +  
 " | CPU\_Usage=" + CPU\_Usage +  
 " | Memory=" + Memory +  
 " | Storage=" + Storage +  
 '|';  
 }  
}  
class Node{  
 Task t;  
 Node next;  
  
 public Node(Task t) {  
 this.t = t;  
 }  
}  
class priorityQueue{  
 Node head;  
 public void enqueue(Task t){  
 Node newNode=new Node(t);  
 if(head==null){  
 head=newNode;  
 }else{  
 Node curr=head;  
 while(curr.next!=null&&curr.next.t.ExecutionTime<= t.ExecutionTime){  
 curr=curr.next;  
 }  
 if(curr.next==null){  
 curr.next=newNode;  
 }else{  
 newNode.next=curr.next;  
 curr.next=newNode;  
 }  
 }  
 }  
 public Task dequeue(){  
 if(head==null){  
 return null;  
 }  
 Task t=head.t;  
 head=head.next;  
  
 return t;  
 }  
 public boolean isEmpty(){  
 return head==null;  
 }  
}

