|  |  |
| --- | --- |
| Name | Nasuha binti Asri |
| Matric Number | 2018256718 |
| Group | M3CS1104J |
| Lecturer’s Name | Madam Rohana binti Ramli |

**Blended Learning 9**

**1. Example 4 Page 121**

**a) Coding**

**Class Node**

public class Node

{

public Object data;

public Node next;

public Node(Object d){

data=d;

}

}

**Class LinkList**

public class LinkList

{

private Node first;

private Node current;

private Node last;

public LinkList(){

first=null;

last=null;

current=null;

}

//No data in the list

public boolean isEmpty(){

return (first==null);

}

public void insertAtFront(Object insertItem){

Node newNode=new Node(insertItem);

if(isEmpty()){

first=newNode;

last=newNode;

}

else{

newNode.next=first;

first=newNode;

}

}

public void insertAtBack(Object insertItem){

Node newNode=new Node(insertItem);

if(isEmpty()){

first=newNode;

last=newNode;

}

else{

last.next=newNode;

last=newNode;

}

}

public Object removeFromFront(){

Object removeItem=null;

if(isEmpty()){

return removeItem;

}

removeItem=first.data;

if(first==last){

first=null;

last=null;

}

else{

first=first.next;

}

return removeItem;

}

public Object removeFromBack(){

Object removeItem=null;

if(isEmpty()){

return removeItem;

}

removeItem=last.data;

if(first==last){

first=null;

last=null;

}

else{

current=first;

while(current.next != last)

current=current.next;

last=current;

last.next=null;

}

return removeItem;

}

public Object getFirst(){

if(isEmpty())

return null;

else{

current=first;

return current.data;

}

}

public Object getNext(){

if(current==last)

return null;

else{

current=current.next;

return current.data;

}

}

}

**Class Stack**

public class Stack extends LinkList

{

public Stack(){

}

public void push(Object elem){

insertAtFront(elem);

}

public Object pop(){

return removeFromFront();

}

public Object peek(){

return getFirst();

}

}

**Class StackApp**

import javax.swing.\*;

public class StackApp

{

public static void main(String[] args){

/\*Declare stack variable\*/

Stack theStack = new Stack();

Stack tempStack = new Stack();

/\*to push data into stack\*/

theStack.push("7");

theStack.push("3");

theStack.push("12");

/\*to get and print data on the top of the stack\*/

System.out.println("DATA ON THE TOP OF THE STACK : ");

System.out.println(theStack.peek());

/\*to remove and print all data in the stack\*/

Object data=null;

System.out.println("DATA IN THE STACK : ");

while(!theStack.isEmpty()){

data = theStack.pop();

System.out.println(data);

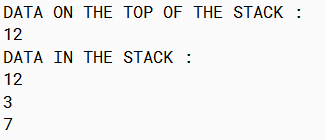
tempStack.push(data);

}

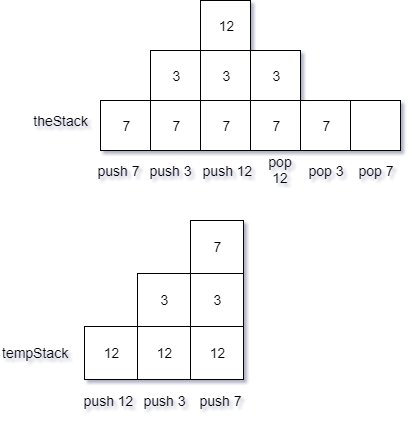
}

}

**b) Output**



**c) Diagram**



**2. Example 5 Page 122**

**a) Coding**

**Class Node**

public class Node

{

public Object data;

public Node next;

public Node(Object d){

data=d;

}

}

**Class LinkList**

public class LinkList

{

private Node first;

private Node current;

private Node last;

public LinkList(){

first=null;

last=null;

current=null;

}

//No data in the list

public boolean isEmpty(){

return (first==null);

}

public void insertAtFront(Object insertItem){

Node newNode=new Node(insertItem);

if(isEmpty()){

first=newNode;

last=newNode;

}

else{

newNode.next=first;

first=newNode;

}

}

public void insertAtBack(Object insertItem){

Node newNode=new Node(insertItem);

if(isEmpty()){

first=newNode;

last=newNode;

}

else{

last.next=newNode;

last=newNode;

}

}

public Object removeFromFront(){

Object removeItem=null;

if(isEmpty()){

return removeItem;

}

removeItem=first.data;

if(first==last){

first=null;

last=null;

}

else{

first=first.next;

}

return removeItem;

}

public Object removeFromBack(){

Object removeItem=null;

if(isEmpty()){

return removeItem;

}

removeItem=last.data;

if(first==last){

first=null;

last=null;

}

else{

current=first;

while(current.next != last)

current=current.next;

last=current;

last.next=null;

}

return removeItem;

}

public Object getFirst(){

if(isEmpty())

return null;

else{

current=first;

return current.data;

}

}

public Object getNext(){

if(current==last)

return null;

else{

current=current.next;

return current.data;

}

}

}

**Class Stack**

public class Stack extends LinkList

{

public Stack(){

}

public void push(Object elem){

insertAtFront(elem);

}

public Object pop(){

return removeFromFront();

}

public Object peek(){

return getFirst();

}

}

**Class StackApp**

import javax.swing.\*;

public class StackAppEX5

{

public static void main(String[] args){

/\*declare stack variable\*/

Stack theStack = new Stack();

Stack tempStack = new Stack();

String num;

/\*Enter string num into stack\*/

for(int i=0; i<3; i++){

num = JOptionPane.showInputDialog("Enter a number: ");

theStack.push(num);

}

/\*display data in the stack\*/

Object data=null;

System.out.println("DATA IN THE STACK: ");

while(!theStack.isEmpty()){

data = theStack.pop();

System.out.println(data);

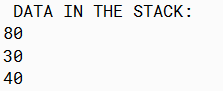
tempStack.push(data);

}

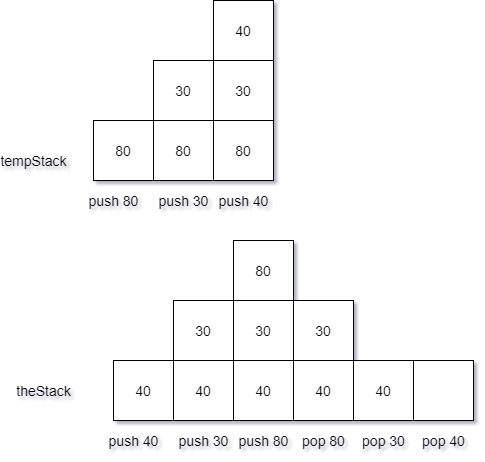
}

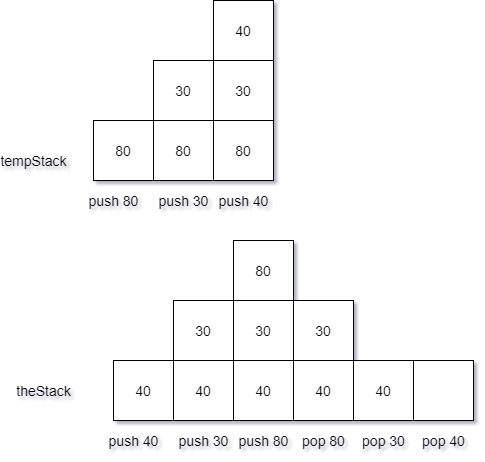
}

**b) Output**



**c) Diagram**





**3. Example 6 Page 122**

**a) Coding**

**Class Node**

public class Node

{

public Object data;

public Node next;

public Node(Object d){

data=d;

}

}

**Class LinkList**

public class LinkList

{

private Node first;

private Node current;

private Node last;

public LinkList(){

first=null;

last=null;

current=null;

}

//No data in the list

public boolean isEmpty(){

return (first==null);

}

public void insertAtFront(Object insertItem){

Node newNode=new Node(insertItem);

if(isEmpty()){

first=newNode;

last=newNode;

}

else{

newNode.next=first;

first=newNode;

}

}

public void insertAtBack(Object insertItem){

Node newNode=new Node(insertItem);

if(isEmpty()){

first=newNode;

last=newNode;

}

else{

last.next=newNode;

last=newNode;

}

}

public Object removeFromFront(){

Object removeItem=null;

if(isEmpty()){

return removeItem;

}

removeItem=first.data;

if(first==last){

first=null;

last=null;

}

else{

first=first.next;

}

return removeItem;

}

public Object removeFromBack(){

Object removeItem=null;

if(isEmpty()){

return removeItem;

}

removeItem=last.data;

if(first==last){

first=null;

last=null;

}

else{

current=first;

while(current.next != last)

current=current.next;

last=current;

last.next=null;

}

return removeItem;

}

public Object getFirst(){

if(isEmpty())

return null;

else{

current=first;

return current.data;

}

}

public Object getNext(){

if(current==last)

return null;

else{

current=current.next;

return current.data;

}

}

}

**Class Stack**

public class Stack extends LinkList

{

public Stack(){

}

public void push(Object elem){

insertAtFront(elem);

}

public Object pop(){

return removeFromFront();

}

public Object peek(){

return getFirst();

}

}

**Class StackNoApp**

import javax.swing.\*;

public class StackNoApp

{

public static void main(String[] args){

Stack myStack = new Stack();

Stack tempStack = new Stack();

String num;

/\*push data into stack\*/

for(int i=0; i<3; i++){

num = JOptionPane.showInputDialog("Enter a number: ");

myStack.push(num);

}

/\*Calculation on items in stack

Find number of items, sum, max, min\*/

int count=0, sum=0, min=99999, max=-99999;

while(!myStack.isEmpty()){

/\*String number from stack masuk dalam Object data\*/

Object data = myStack.pop();

count++;

/\*Change data from string to int\*/

int no = Integer.parseInt(data.toString());

sum = sum + no;

if(no > max){

max = no;

}

if(no < min){

min = no;

}

/\*Push data into tempStack\*/

tempStack.push(data);

}

double avg = sum / count;

/\*Print result\*/

JOptionPane.showMessageDialog(null, "There are " + count + " items in the stack");

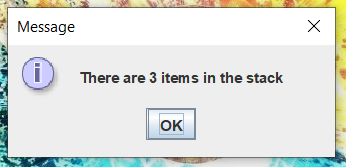
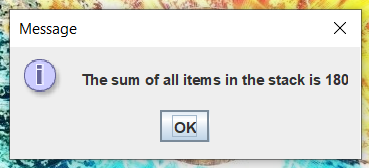
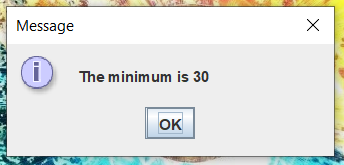
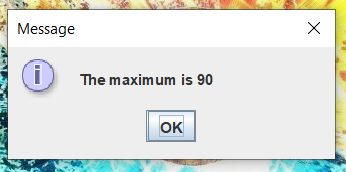
JOptionPane.showMessageDialog(null, "The sum of all items in the stack is " + sum);

JOptionPane.showMessageDialog(null, "The minimum is " + min);

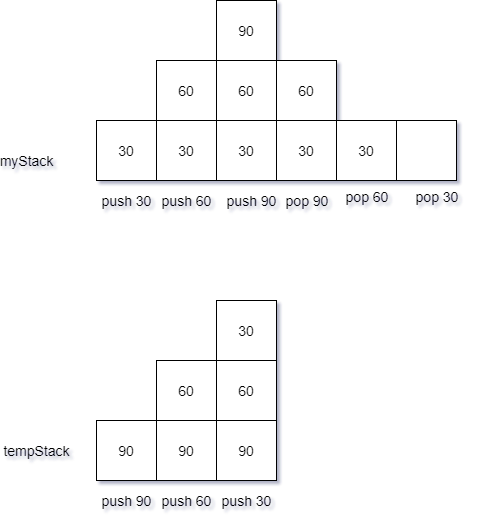
JOptionPane.showMessageDialog(null, "The maximum is " + max);

}

}

**b) Output**

**c) Diagram**



**4. Example 7 Page 124**

**a) Coding**

**Class Node**

public class Node

{

public Object data;

public Node next;

public Node(Object d){

data=d;

}

}

**Class LinkList**

public class LinkList

{

private Node first;

private Node current;

private Node last;

public LinkList(){

first=null;

last=null;

current=null;

}

//No data in the list

public boolean isEmpty(){

return (first==null);

}

public void insertAtFront(Object insertItem){

Node newNode=new Node(insertItem);

if(isEmpty()){

first=newNode;

last=newNode;

}

else{

newNode.next=first;

first=newNode;

}

}

public void insertAtBack(Object insertItem){

Node newNode=new Node(insertItem);

if(isEmpty()){

first=newNode;

last=newNode;

}

else{

last.next=newNode;

last=newNode;

}

}

public Object removeFromFront(){

Object removeItem=null;

if(isEmpty()){

return removeItem;

}

removeItem=first.data;

if(first==last){

first=null;

last=null;

}

else{

first=first.next;

}

return removeItem;

}

public Object removeFromBack(){

Object removeItem=null;

if(isEmpty()){

return removeItem;

}

removeItem=last.data;

if(first==last){

first=null;

last=null;

}

else{

current=first;

while(current.next != last)

current=current.next;

last=current;

last.next=null;

}

return removeItem;

}

public Object getFirst(){

if(isEmpty())

return null;

else{

current=first;

return current.data;

}

}

public Object getNext(){

if(current==last)

return null;

else{

current=current.next;

return current.data;

}

}

}

**Class Stack**

public class Stack extends LinkList

{

public Stack(){

}

public void push(Object elem){

insertAtFront(elem);

}

public Object pop(){

return removeFromFront();

}

public Object peek(){

return getFirst();

}

}

**Class Employee**

import javax.swing.\*;

public class Employee

{

private String empID;

private String empName;

private int empAge;

private double empSalary;

public Employee(){

empID=" ";

empName= " ";

empAge=0;

empSalary=0.0;

}

public Employee(String empID, String empName, int empAge, double empSlry){

this.empID=empID;

this.empName=empName;

this.empAge=empAge;

this.empSalary=empSlry;

}

public void setEmployee(String ID, String Name, int age, double salary){

empID=ID;

empName=Name;

empAge=age;

empSalary=salary;

}

public String getID(){

return empID;

}

public String getName(){

return empName;

}

public int getAge(){

return empAge;

}

public double getSalary(){

return empSalary;

}

public String toString(){

return ("ID: " + empID + "\nName: " + empName + "\nAge: " + empAge + "\nSalary: RM " + empSalary);

}

}

**Class StackApp**

import javax.swing.\*;

public class StackApp

{

public static void main(String[] args)

{

Stack theStack = new Stack();

Stack tempStack = new Stack();

/\*to input 5 employees into the list\*/

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

String eID = JOptionPane.showInputDialog("Enter employee ID: ");

String eName = JOptionPane.showInputDialog("Enter employee name: ");

String eAge = JOptionPane.showInputDialog("Enter employee age: ");

String eSalary = JOptionPane.showInputDialog("Enter employee salary: ");

int age = Integer.parseInt(eAge);

double salary = Double.parseDouble(eSalary);

Employee E = new Employee(eID, eName, age, salary);

theStack.push(E);

}

/\*to display all employees in the stack\*/

Object data;

Employee E;

while(!theStack.isEmpty()){

/\*data in stack put into Object data\*/

data = theStack.pop();

/\*Declare data as Employee\*/

E = (Employee)data;

System.out.println(E.toString());

/\*put into temporary stack\*/

tempStack.push(E);

}

/\*Transfer all data from temp stack into original stack\*/

while(!tempStack.isEmpty()){

data = tempStack.pop();

E = (Employee)data;

theStack.push(E);

}

int numAge=0, num2000=0;

while(!theStack.isEmpty()){

data = theStack.pop();

E = (Employee)data;

if(E.getAge() > 40){

numAge++;

}

if(E.getSalary() > 2000){

num2000++;

}

tempStack.push(E);

}

/\*Display result\*/

System.out.println("\nThe number of employee that their age is above 40: " + numAge);

System.out.println("The number of employee that salary is above RM2000: " + num2000);

double maxSalary=-99999.99, minSalary=99999.99;

while(!tempStack.isEmpty()){

data = tempStack.pop();

E = (Employee)data;

if(E.getSalary() > maxSalary){

maxSalary = E.getSalary();

}

else{

minSalary = E.getSalary();

}

theStack.push(E);

}

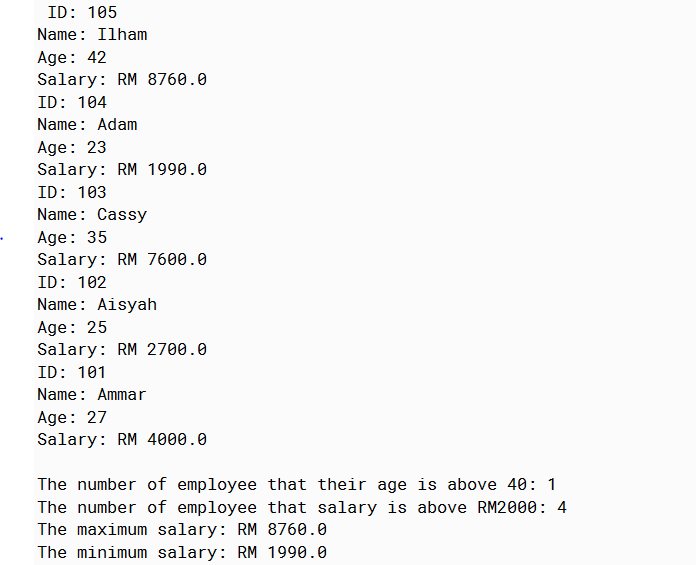
System.out.println("The maximum salary: RM " + maxSalary);

System.out.println("The minimum salary: RM " + minSalary);

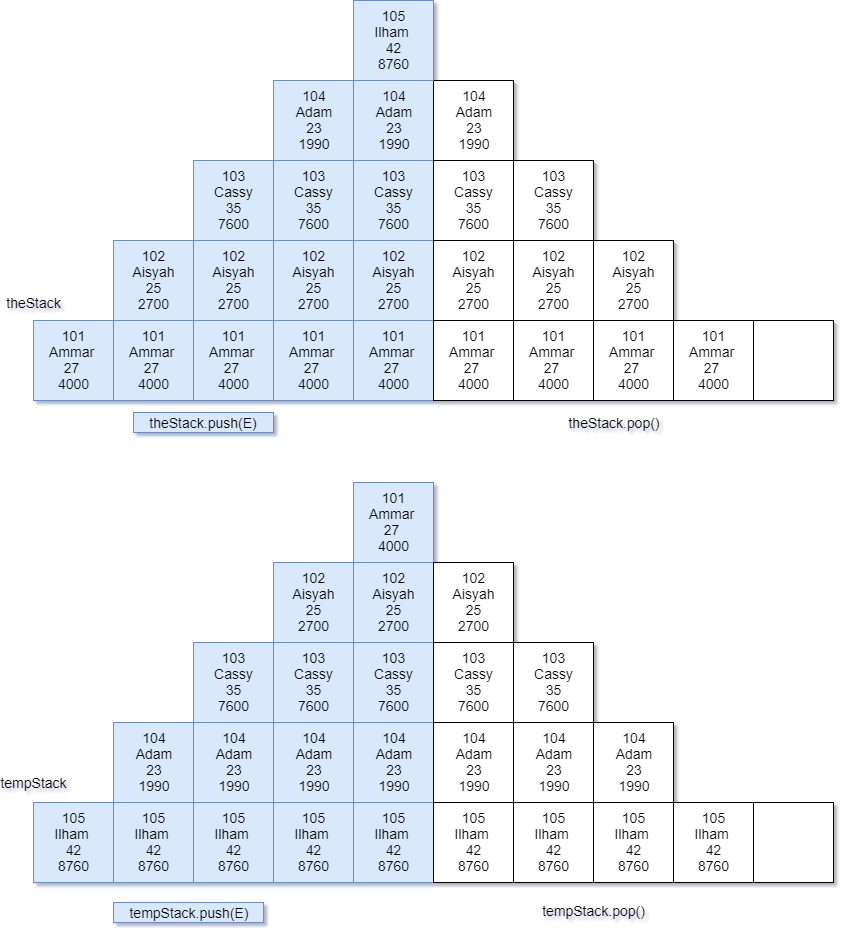
}

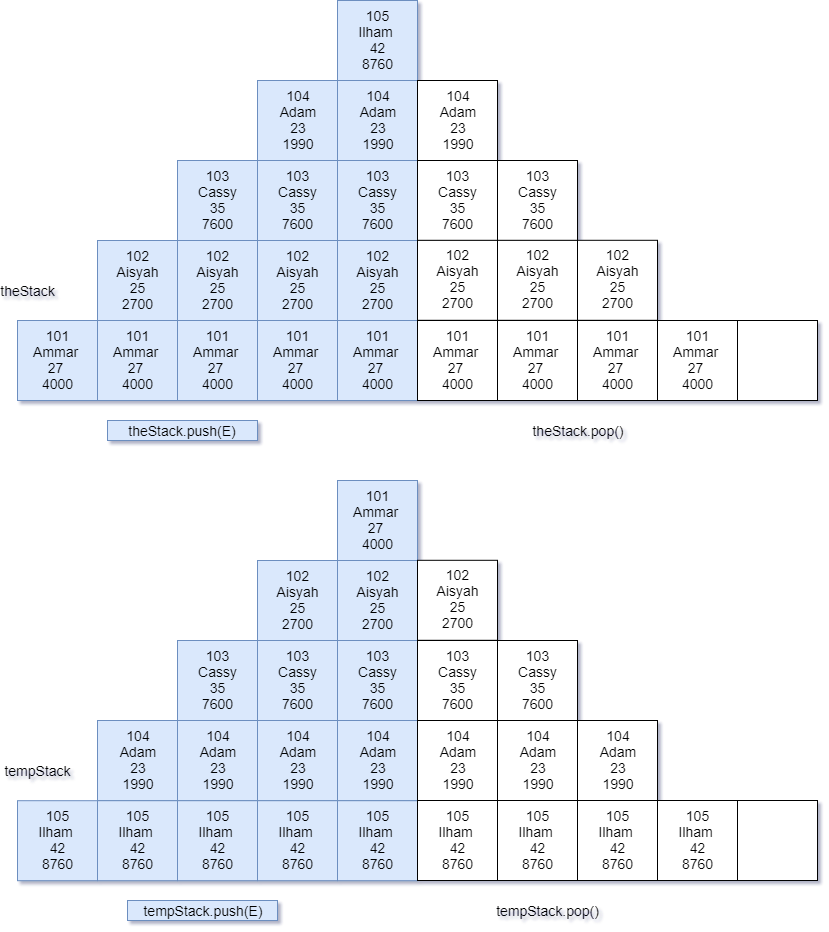
}

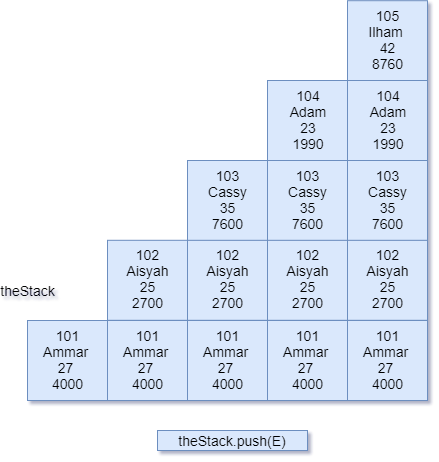
**b) Output**



**c) Diagram**

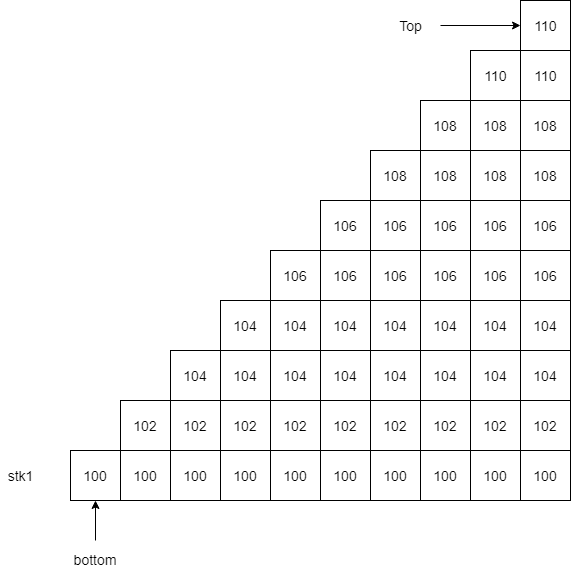






**5. a) December 2019, Question 5, Page 127**

a)



b) Object data=null;

while(!stk2.isEmpty()){

data = stk2.pop();

temp.push(data);

stk2 = temp;

}

**b) Examination June 2019, Question 6d**

Object data=null;

while(!stack1.isEmpty()){

data = stack1.pop();

int num = Integer.parseInt(data.toString());

if(num %2 != 0){

tempStack.push(data);

}

}

**c) Examination Jan 2018, Question 6**

a) Object data=null;

Book B;

while(!bookStack.isEmpty()){

data = bookStack.pop();

B = (Book)data;

if(B.getSerialNum().charAt(0)=='A'){

kidStack.push(data);

}

else if(B.getSerialNum().charAt(0)=='B'){

adultStack.push(data);

}

else {

tempStack.push(data);

}

}

b)

