NAMES:MUGIRANEZA H. yves reg:217129471

DATA STRUCTURE AND ALHORTHM ASSIGNMENT

1. algorithm to find the largest of a set of numbers.you do not know the number of numbersSSS

Input: The sequence A0, A1,...,An-1 and the length n of the sequence

Output: large, the largest element in this sequence

procedure: large( A[]:integer array, n:integer)

i:integer;

i := 1;

large := A[0];

while i < n do

if A[i] > large then

large := A[i];

i := i + 1;

return(large)

2. algorithm in pseudocode that finds the average of (n) numbers

main()

start

input number, sum ← 0, average ← 0 //declare average as float type

Write "please input size of array"

Read number

declare array[number]

Write "Enter numbers in array"

for i←0; i<n; i←i+1 do

sum ← sum+array[i]

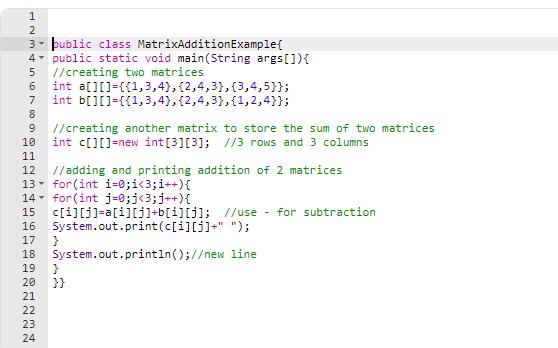
average ← sum/number

Print average

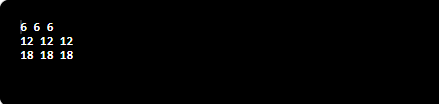
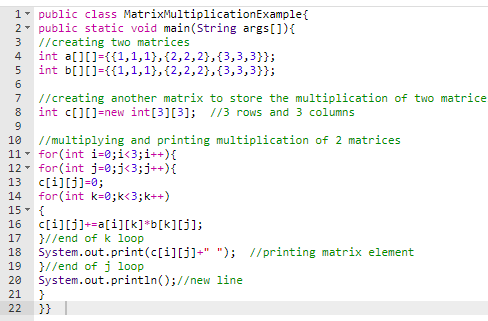
End for

End

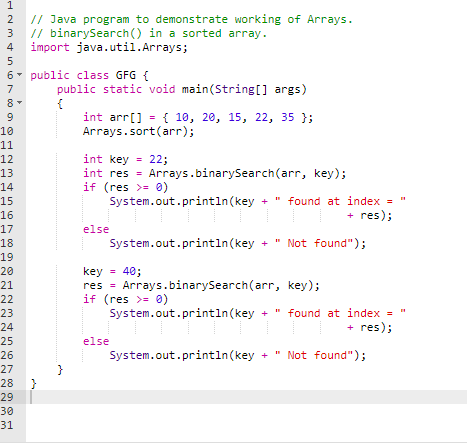
3.Matrix addition



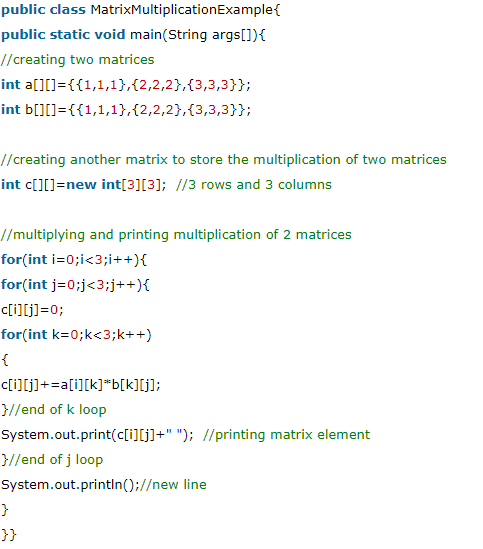
4. Multiplication of matrix



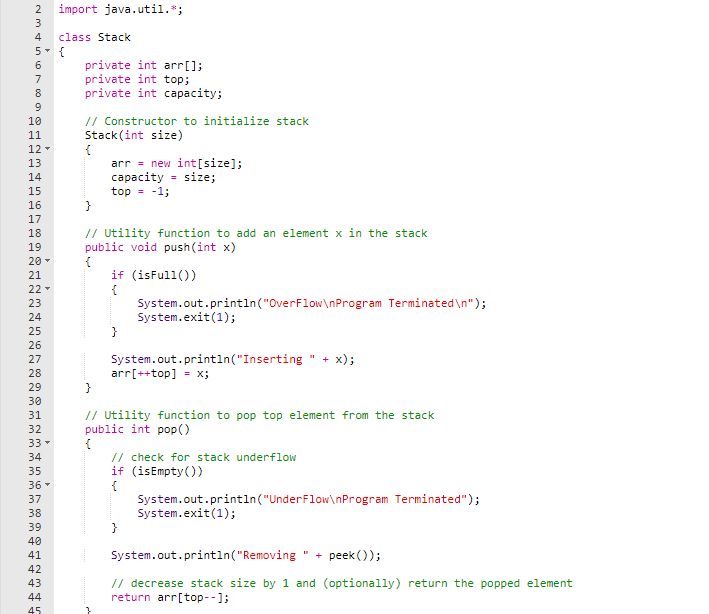
5. binary search

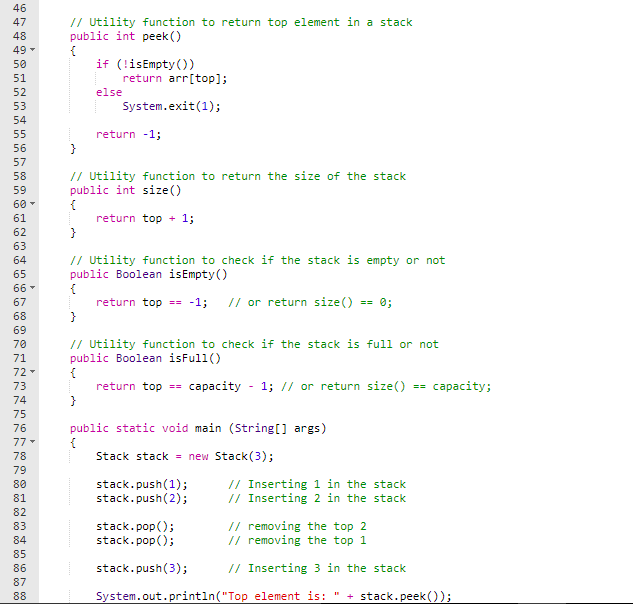
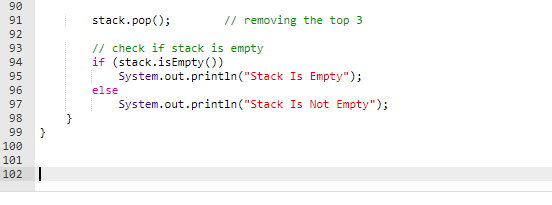


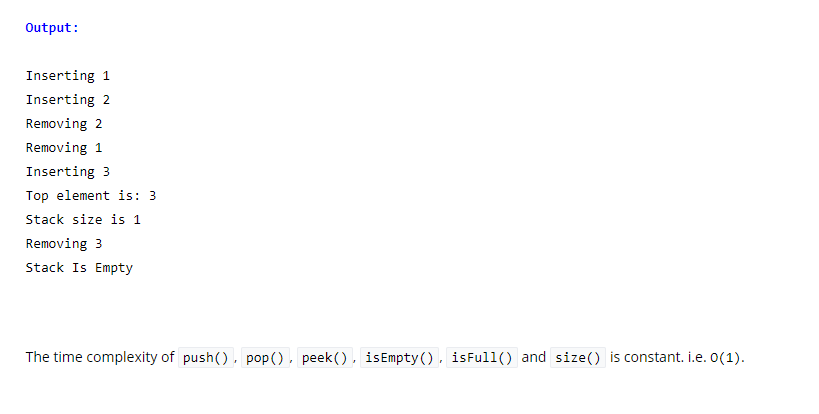
6. Linear search



7.stack





8. QUEUE

1. */\**
2. *\* Java Program to Implement Queue*
3. *\*/*
5. **import** java.util.\*;
7. */\* Class arrayQueue \*/*
8. **class** arrayQueue
9. {
10. **protected** **int** Queue[] ;
11. **protected** **int** front, rear, size, len;
13. */\* Constructor \*/*
14. **public** arrayQueue(**int** n)
15. {
16. size = n;
17. len = 0;
18. Queue = **new** **int**[size];
19. front = -1;
20. rear = -1;
21. }
22. */\* Function to check if queue is empty \*/*
23. **public** **boolean** isEmpty()
24. {
25. **return** front == -1;
26. }
27. */\* Function to check if queue is full \*/*
28. **public** **boolean** isFull()
29. {
30. **return** front==0 && rear == size -1 ;
31. }
32. */\* Function to get the size of the queue \*/*
33. **public** **int** getSize()
34. {
35. **return** len ;
36. }
37. */\* Function to check the front element of the queue \*/*
38. **public** **int** peek()
39. {
40. **if** (isEmpty())
41. **throw** **new** NoSuchElementException("Underflow Exception");
42. **return** Queue[front];
43. }
44. */\* Function to insert an element to the queue \*/*
45. **public** **void** insert(**int** i)
46. {
47. **if** (rear == -1)
48. {
49. front = 0;
50. rear = 0;
51. Queue[rear] = i;
52. }
53. **else** **if** (rear + 1 >= size)
54. **throw** **new** IndexOutOfBoundsException("Overflow Exception");
55. **else** **if** ( rear + 1 < size)
56. Queue[++rear] = i;
57. len++ ;
58. }
59. */\* Function to remove front element from the queue \*/*
60. **public** **int** remove()
61. {
62. **if** (isEmpty())
63. **throw** **new** NoSuchElementException("Underflow Exception");
64. **else**
65. {
66. len-- ;
67. **int** ele = Queue[front];
68. **if** ( front == rear)
69. {
70. front = -1;
71. rear = -1;
72. }
73. **else**
74. front++;
75. **return** ele;
76. }
77. }
78. */\* Function to display the status of the queue \*/*
79. **public** **void** display()
80. {
81. System.out.print("**\n**Queue = ");
82. **if** (len == 0)
83. {
84. System.out.print("Empty**\n**");
85. **return** ;
86. }
87. **for** (**int** i = front; i <= rear; i++)
88. System.out.print(Queue[i]+" ");
89. System.out.println();
90. }
91. }
93. */\* Class QueueImplement \*/*
94. **public** **class** QueueImplement
95. {
96. **public** **static** **void** main(String[] args)
97. {
98. Scanner scan = **new** Scanner(System.in);
100. System.out.println("Array Queue Test**\n**");
101. System.out.println("Enter Size of Integer Queue ");
102. **int** n = scan.nextInt();
103. */\* creating object of class arrayQueue \*/*
104. arrayQueue q = **new** arrayQueue(n);
105. */\* Perform Queue Operations \*/*
106. **char** ch;
107. **do**{
108. System.out.println("**\n**Queue Operations");
109. System.out.println("1. insert");
110. System.out.println("2. remove");
111. System.out.println("3. peek");
112. System.out.println("4. check empty");
113. System.out.println("5. check full");
114. System.out.println("6. size");
115. **int** choice = scan.nextInt();
116. **switch** (choice)
117. {
118. **case** 1 :
119. System.out.println("Enter integer element to insert");
120. **try**
121. {
122. q.insert( scan.nextInt() );
123. }
124. **catch**(Exception e)
125. {
126. System.out.println("Error : " +e.getMessage());
127. }
128. **break**;
129. **case** 2 :
130. **try**
131. {
132. System.out.println("Removed Element = "+q.remove());
133. }
134. **catch**(Exception e)
135. {
136. System.out.println("Error : " +e.getMessage());
137. }
138. **break**;
139. **case** 3 :
140. **try**
141. {
142. System.out.println("Peek Element = "+q.peek());
143. }
144. **catch**(Exception e)
145. {
146. System.out.println("Error : "+e.getMessage());
147. }
148. **break**;
149. **case** 4 :
150. System.out.println("Empty status = "+q.isEmpty());
151. **break**;
152. **case** 5 :
153. System.out.println("Full status = "+q.isFull());
154. **break**;
155. **case** 6 :
156. System.out.println("Size = "+ q.getSize());
157. **break**;
158. **default** : System.out.println("Wrong Entry **\n** ");
159. **break**;
160. }
161. */\* display Queue \*/*
162. q.display();
163. System.out.println("**\n**Do you want to continue (Type y or n) **\n**");
164. ch = scan.next().charAt(0);
166. } **while** (ch == 'Y'|| ch == 'y');
167. }
168. }

OUTPUT

Array Queue Test

Enter Size of Integer Queue

5

Queue Operations

1. insert

2. remove

3. peek

4. check empty

5. check full

6. **size**

4

Empty status = **true**

Queue = Empty

Do you want to **continue** **(**Type y or n**)**

y

Queue Operations

1. insert

2. remove

3. peek

4. check empty

5. check full

6. **size**

1

Enter integer element to insert

24

Queue = 24

Do you want to **continue** **(**Type y or n**)**

y

Queue Operations

1. insert

2. remove

3. peek

4. check empty

5. check full

6. **size**

1

Enter integer element to insert

6

Queue = 24 6

Do you want to **continue** **(**Type y or n**)**

y

Queue Operations

1. insert

2. remove

3. peek

4. check empty

5. check full

6. **size**

1

Enter integer element to insert

16

Queue = 24 6 16

Do you want to **continue** **(**Type y or n**)**

y

Queue Operations

1. insert

2. remove

3. peek

4. check empty

5. check full

6. **size**

1

Enter integer element to insert

19

Queue = 24 6 16 19

Do you want to **continue** **(**Type y or n**)**

y

Queue Operations

1. insert

2. remove

3. peek

4. check empty

5. check full

6. **size**

1

Enter integer element to insert

32

Queue = 24 6 16 19 32

Do you want to **continue** **(**Type y or n**)**

y

Queue Operations

1. insert

2. remove

3. peek

4. check empty

5. check full

6. **size**

1

Enter integer element to insert

14

Error : Overflow Exception

Queue = 24 6 16 19 32

Do you want to **continue** **(**Type y or n**)**

y

Queue Operations

1. insert

2. remove

3. peek

4. check empty

5. check full

6. **size**

5

Full status = **true**

Queue = 24 6 16 19 32

Do you want to **continue** **(**Type y or n**)**

y

Queue Operations

1. insert

2. remove

3. peek

4. check empty

5. check full

6. **size**

3

Peek Element = 24

Queue = 24 6 16 19 32

Do you want to **continue** **(**Type y or n**)**

y

Queue Operations

1. insert

2. remove

3. peek

4. check empty

5. check full

6. **size**

2

Removed Element = 24

Queue = 6 16 19 32

Do you want to **continue** **(**Type y or n**)**

y

Queue Operations

1. insert

2. remove

3. peek

4. check empty

5. check full

6. **size**

2

Removed Element = 6

Queue = 16 19 32

Do you want to **continue** **(**Type y or n**)**

y

Queue Operations

1. insert

2. remove

3. peek

4. check empty

5. check full

6. **size**

6

Size = 3

Queue = 16 19 32

Do you want to **continue** **(**Type y or n**)**

y

Queue Operations

1. insert

2. remove

3. peek

4. check empty

5. check full

6. **size**

3

Peek Element = 16

Queue = 16 19 32

Do you want to **continue** **(**Type y or n**)**

y

Queue Operations

1. insert

2. remove

3. peek

4. check empty

5. check full

6. **size**

2

Removed Element = 16

Queue = 19 32

Do you want to **continue** **(**Type y or n**)**

y

Queue Operations

1. insert

2. remove

3. peek

4. check empty

5. check full

6. **size**

2

Removed Element = 19

Queue = 32

Do you want to **continue** **(**Type y or n**)**

y

Queue Operations

1. insert

2. remove

3. peek

4. check empty

5. check full

6. **size**

2

Removed Element = 32

Queue = Empty

Do you want to **continue** **(**Type y or n**)**

y

Queue Operations

1. insert

2. remove

3. peek

4. check empty

5. check full

6. **size**

2

Error : Underflow Exception

Queue = Empty

Do you want to **continue** **(**Type y or n**)**

y

Queue Operations

1. insert

2. remove

3. peek

4. check empty

5. check full

6. **size**

4

Empty status = **true**

Queue = Empty

Do you want to **continue** **(**Type y or n**)**

n