Ananya Sharma **102215299** **2NC8**

Lab Assignment 5

UCS 406 Data Structures and Algorithms

```
/*

Roll Number:102215255
Name: Diya Goyal
Description: 1Given array A[] with sliding window of size w which is moving from the very left of the
array to the very right. Assume that we can only see the w numbers in the window. Each time the
sliding window moves rightwards by one position. For example: The array is [1 3 -1 -3 5 3 6 7], and w
is 3.
Acknowledgement: GeeksforGeeks and took help from friends
*/
#include <bits/stdc++.h>
using namespace std;
void printKMax(int arr[], int N, int K)
{
   int j, max;

   for (int i = 0; i <= N - K; i++) {
      max = arr[i];

      for (j = 1; j < K; j++) {
         if (arr[i + j] > max)
            max = arr[i + j];
      }
      cout << "Max:" << max <<endl;
   }
}
int main()
{
   int arr[] = { 1,3,-1,-3,5,3,6,7 };
   int N = sizeof(arr) / sizeof(arr[0]);
   int K = 3;
   printKMax(arr, N, K);
   return 0;
}
```

Output:

```
Max:3
Max:3
Max:5
Max:5
Max:6
Max:7
```

```
/*
Roll Number:102215255
Name: Diya Goyal
Description:2.  Given a Linked list, List1 = {A 1 , A 2 , . . . A n–1 ; A n ) with data, write a program to
re-order it to {A 1 , A n , A 2 , A n–1 …} without using any extra space.
Acknowledgement: GeeksforGeeks and discussion with friends
*/
#include <bits/stdc++.h>
using namespace std;

struct Node {
   int data;
   struct Node* next;
};
Node* newNode(int key)
{
   Node* temp = new Node;
   temp->data = key;
   temp->next = NULL;
   return temp;
}
void reverselist(Node** head)
{
   Node *prev = NULL, *curr = *head, *next;

   while (curr) {
      next = curr->next;
      curr->next = prev;
      prev = curr;
      curr = next;
   }
   *head = prev;
}

void printlist(Node* head)
{
   while (head != NULL) {
      cout << head->data << " ";
      if (head->next)
         cout << "-> ";
      head = head->next;
   }
   cout << endl;
}

void rearrange(Node** head)
{
   Node *slow = *head, *fast = slow->next;
```

```
    while (fast && fast->next) {
        slow = slow->next;
        fast = fast->next->next;
    }
    Node* head1 = *head;
    Node* head2 = slow->next;
    slow->next = NULL;
    reverselist(&head2);
    *head = newNode(0);
    Node* curr = *head;
    while (head1 || head2) {
        if (head1) {
            curr->next = head1;
            curr = curr->next;
            head1 = head1->next;
        }
        if (head2) {
            curr->next = head2;
            curr = curr->next;
            head2 = head2->next;
        }
    }
    *head = (*head)->next;
}

int main()
{
    Node* head = newNode(11);
    head->next = newNode(12);
    head->next->next = newNode(15);
    head->next->next->next = newNode(19);
    head->next->next->next->next = newNode(23);

    printlist(head);
    rearrange(&head);
    printlist(head);
    return 0;
}
```

Output:

```
11 -> 12 -> 15 -> 19 -> 23
11 -> 23 -> 12 -> 19 -> 15

--------------------------------
Process exited after 0.02513 seconds with
Press any key to continue . . .
```

/*

Roll Number: 102215255
Name: Diya Goyal
Description:3.

- What is the value of the following postfix expression?
  54 + 7 4 - * 9 / 35 15 + +
- Use the conversion algorithm to change the following infix expression into postfix using stack. Show each step using a tabular approach.
  (A * B - (C - D)) / (E + F)
- Also write the program to perform this conversion. Also write a program to evaluate the postfix expression.

Acknowledgement: NA
*/

- **Value of the postfix expression**
  54 6 + --> 60
  7 4 - --> 3
  60 * 3 --> 180
  9 / --> 20
  35 15 + --> 50
  180 + 50 + --> 230

- **Conversion from Infix to Postfix:**

Symbol	Stack	Postfix
(	(	*
A	(A	*
*	(A	- *
B	(A B	- *
)	(A B	- *
-	(A B *	-
(	(A B *	-
C	(A B * C	-
-	(A B * C	- -
D	(A B * C D	- -
)	(A B * C D -	-
)	(A B * C D -	-
/	(A B * C D --	/
(	(A B * C D --	/
E	(A B * C D -- E	/
+	(A B * C D -- E	/
F	(A B * C D -- E F	/+
)	(A B * C D -- E F +	/

Postfix expression is **AB*CD--EF+/**

```
/* Acknowledgement: GeeksforGeeks*/
#include <bits/stdc++.h>
using namespace std;

int prec(char c) {
   if (c == '^')
      return 3;
   else if (c == '/' || c == '*')
      return 2;
   else if (c == '+' || c == '-')
      return 1;
   else
      return -1;
}

char associativity(char c) {
   if (c == '^')
      return 'R';
   return 'L';
}

void infixToPostfix(string s) {
   stack<char> st;
   string result;

   for (int i = 0; i < s.length(); i++) {
      char c = s[i];

      if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9'))
         result += c;

      else if (c == '(')
         st.push('(');

      else if (c == ')') {
         while (st.top() != '(') {
            result += st.top();
            st.pop();
         }
         st.pop();
      }

      else {
         while (!st.empty() && prec(s[i]) < prec(st.top()) ||
               !st.empty() && prec(s[i]) == prec(st.top()) &&
               associativity(s[i]) == 'L') {
            result += st.top();
            st.pop();
```

```
        }
        st.push(c);
      }
    }

    while (!st.empty()) {
      result += st.top();
      st.pop();
    }

    cout << result << endl;
}

int main() {
    string exp = "(A*B-(C-D))/(E+F)";
    infixToPostfix(exp);

    return 0;
}
```

Output:

```
AB*CD--EF+/

--------------------------------
Process exited after 0.02602 secc
Press any key to continue . . .
```

```
/*
Roll Number: 102215255
Name: Diya Goyal
Description:4. Write a program to perform Parenthesis matching in an expression.
Acknowledgement: ChatGPT
*/
#include <iostream>
#include <stack>
#include <string>
using namespace std;

bool isMatching(char opening, char closing) {
    return (opening == '(' && closing == ')') ||
        (opening == '[' && closing == ']') ||
        (opening == '{' && closing == '}');
}
```

```
bool isBalanced(const std::string& expression) {
   stack<char> stack;

   for (size_t i = 0; i < expression.length(); ++i) {
   char c = expression[i];
      if (c == '(' || c == '[' || c == '{') {
         stack.push(c);
      } else if (c == ')' || c == ']' || c == '}') {
         if (stack.empty() || !isMatching(stack.top(), c)) {
            return false;
         }
         stack.pop();
      }
   }
   return stack.empty();
}

int main() {
   string expression;
   cout << "Enter the expression: ";
   getline(std::cin, expression);

   if (isBalanced(expression)) {
      cout << "Parentheses are balanced." << endl;
   } else {
      cout << "Parentheses are not balanced." << endl;
   }
   return 0;
}
```

Output:

```
Enter the expression: (A * B - (C - D)) / (E + F)
Parentheses are balanced.

--------------------------------
Process exited after 18.89 seconds with return value 0
Press any key to continue . . .
```