Q1.

// C++ Program infix to postfix expression using stack (array) in data structure

#include<iostream>

#include<string>

#define MAX 20

using namespace std;

char stk[20];

int top=-1;

// Push function here, inserts value in stack and increments stack top by 1

void push(char oper)

{

if(top==MAX-1)

{

cout<<"stackfull!!!!";

}

else

{

top++;

stk[top]=oper;

}

}

// Function to remove an item from stack. It decreases top by 1

char pop()

{

char ch;

if(top==-1)

{

cout<<"stackempty!!!!";

}

else

{

ch=stk[top];

stk[top]='\0';

top--;

return(ch);

}

return 0;

}

int priority ( char alpha )

{

if(alpha == '+' || alpha =='-')

{

return(1);

}

if(alpha == '\*' || alpha =='/')

{

return(2);

}

if(alpha == '$')

{

return(3);

}

return 0;

}

string convert(string infix)

{

int i=0;

string postfix = "";

while(infix[i]!='\0')

{

if(infix[i]>='a' && infix[i]<='z'|| infix[i]>='A'&& infix[i]<='Z')

{

postfix.insert(postfix.end(),infix[i]);

i++;

}

else if(infix[i]=='(' || infix[i]=='{' || infix[i]=='[')

{

push(infix[i]);

i++;

}

else if(infix[i]==')' || infix[i]=='}' || infix[i]==']')

{

if(infix[i]==')')

{

while(stk[top]!='(')

{ postfix.insert(postfix.end(),pop());

}

pop();

i++;

}

if(infix[i]==']')

{

while(stk[top]!='[')

{

postfix.insert(postfix.end(),pop());

}

pop();

i++;

}

if(infix[i]=='}')

{

while(stk[top]!='{')

{

postfix.insert(postfix.end(),pop());

}

pop();

i++;

}

}

else

{

if(top==-1)

{

push(infix[i]);

i++;

}

else if( priority(infix[i]) <= priority(stk[top])) {

postfix.insert(postfix.end(),pop());

while(priority(stk[top]) == priority(infix[i])){

postfix.insert(postfix.end(),pop());

if(top < 0) {

break;

}

}

push(infix[i]);

i++;

}

else if(priority(infix[i]) > priority(stk[top])) {

push(infix[i]);

i++;

}

}

}

while(top!=-1)

{

postfix.insert(postfix.end(),pop());

}

cout<<"The converted postfix string is : "<<postfix; //it will print postfix conversion

return postfix;

}

int main()

{

int cont;

string infix, postfix;

cout<<"\nEnter the infix expression : "; //enter the expression

cin>>infix;

postfix = convert(infix);

return 0;

}

Text

Description automatically generated

Q2.

// C++ implementation to insert value in sorted way

// in a sorted doubly linked list

#include <bits/stdc++.h>

using namespace std;

struct Node {

int data;

struct Node\* prev, \*next;

};

struct Node\* getNode(int data)

{

struct Node\* newNode =

(struct Node\*)malloc(sizeof(struct Node));

newNode->data = data;

newNode->prev = newNode->next = NULL;

return newNode;

}

void sortedInsert(struct Node\*\* head\_ref, struct Node\* newNode)

{

struct Node\* current;

if (\*head\_ref == NULL)

\*head\_ref = newNode;

else if ((\*head\_ref)->data >= newNode->data) {

newNode->next = \*head\_ref;

newNode->next->prev = newNode;

\*head\_ref = newNode;

}

else {

current = \*head\_ref;

while (current->next != NULL &&

current->next->data < newNode->data)

current = current->next;

newNode->next = current->next;

if (current->next != NULL)

newNode->next->prev = newNode;

current->next = newNode;

newNode->prev = current;

}

}

void printList(struct Node\* head)

{

while (head != NULL) {

cout << head->data << " ";

head = head->next;

}

}

int main()

{

struct Node\* head = NULL;

struct Node\* new\_node = getNode(6);

sortedInsert(&head, new\_node);

new\_node = getNode(4);

sortedInsert(&head, new\_node);

new\_node = getNode(1);

sortedInsert(&head, new\_node);

new\_node = getNode(10);

sortedInsert(&head, new\_node);

new\_node = getNode(5);

sortedInsert(&head, new\_node);

cout << "Current Doubly Linked List \n";

printList(head);

new\_node = getNode(9);

sortedInsert(&head, new\_node);

cout << "\nAfter adding 9 Doubly Linked List becomes \n";

printList(head);

return 0;

}

Text

Description automatically generated

Q3. // C++ program to find first and last occurrences of

// a number in a given sorted array

#include <bits/stdc++.h>

using namespace std;

int first(int arr[], int low, int high, int x, int n)

{

if (high >= low) {

int mid = low + (high - low) / 2;

if ((mid == 0 || x > arr[mid - 1]) && arr[mid] == x)

return mid;

else if (x > arr[mid])

return first(arr, (mid + 1), high, x, n);

else

return first(arr, low, (mid - 1), x, n);

}

return -1;

}

int last(int arr[], int low, int high, int x, int n)

{

if (high >= low) {

int mid = low + (high - low) / 2;

if ((mid == n - 1 || x < arr[mid + 1]) && arr[mid] == x)

return mid;

else if (x < arr[mid])

return last(arr, low, (mid - 1), x, n);

else

return last(arr, (mid + 1), high, x, n);

}

return -1;

}

int main()

{

int arr[] = { 1, 2, 2, 2, 2, 3, 4, 7, 8, 8 };

int n = sizeof(arr) / sizeof(int);

int x = 8;

printf("First Occurrence = %d\t",

first(arr, 0, n - 1, x, n));

printf("\nLast Occurrence = %d\n",

last(arr, 0, n - 1, x, n));

return 0;

}

Graphical user interface, application

Description automatically generated

Q4.

// C++ implementation of the above approach

#include <bits/stdc++.h>

using namespace std;

// Function to find out the number of

// candies every person received

void candies(int n, int k)

{

// Count number of complete turns

int count = 0;

// Get the last term

int ind = 1;

// Stores the number of candies

int arr[k];

memset(arr, 0, sizeof(arr));

int low = 0, high = n;

// Do a binary search to find the number whose

// sum is less than N.

while (low <= high) {

// Get mide

int mid = (low + high) >> 1;

int sum = (mid \* (mid + 1)) >> 1;

// If sum is below N

if (sum <= n) {

// Find number of complete turns

count = mid / k;

// Right halve

low = mid + 1;

}

else {

// Left halve

high = mid - 1;

}

}

// Last term of last complete series

int last = (count \* k);

// Subtract the sum till

n -= (last \* (last + 1)) / 2;

int i = 0;

// First term of incomplete series

int term = (count \* k) + 1;

while (n) {

if (term <= n) {

arr[i++] = term;

n -= term;

term++;

}

else {

arr[i] += n;

n = 0;

}

}

// Count the total candies

for (int i = 0; i < k; i++)

arr[i] += (count \* (i + 1))

+ (k \* (count \* (count - 1)) / 2);

// Print the total candies

for (int i = 0; i < k; i++)

cout << arr[i] << " ";

}

// Driver Code

int main()

{

int n = 7, k = 4;

candies(n, k);

return 0;

}

Graphical user interface, text, application

Description automatically generated