Name-Jai Shivam Chaudhary

**ASSIGNMENT-1**

1.**C++ Program to count the total number of characters in a string**

//count the total number of characters in a string

#include<iostream>

#include<string.h>

using namespace std;

main()

{

int count=0;

char s[40];

cin.getline(s,40);

for(int i=0;i<strlen(s);i++)

{

if(s[i]!=0)

{

count++;

}

}

cout<<count;

}

2**.C++ Program to count the total number of punctuation characters exists in a String**

#include<iostream>

#include<string.h>

using namespace std;

main()

{

cout<<"enter the length ";

int n;

cin>>n;

cout<<"enter the string with puntuation "<<endl;

char c[100];

cin>>c;

int count=0;

int i=0;

while(i<=n)

{

if(c[i]==','||c[i]=='.'||c[i]=='!'||c[i]==':'||c[i]==';')

{

count++;

}

i++;

}

cout<<"number of puntuation "<<count;

}

16) C++ Program to find Reverse of the string

#include<iostream>

#include<string.h>

using namespace std;

main()

{

char c[80];

cout<<"enter the string to reverse ";

cin.getline(c,80);

for(int i=strlen(c);i>=0;i--)

{

cout<<c[i];

}

}

**4) C++ Program to count the total number of vowels and consonants in a string**

#include<iostream>

using namespace std;

main()

{

int ner=0;

char c[18];

cin.getline(c,18);

int count=0;

int i=0;

while(i<=18)

{

if(c[i]=='A'||c[i]=='E'||c[i]=='I'||c[i]=='O'||c[i]=='U')

{

count++;

}

else

{

ner++;

}

i++;

}

cout<<"number of vowel "<<count<<endl;

cout<<"number of consonant "<<ner<<endl;

}

**C++ Program to find the longest repeating sequence in a string**

//longest repeating

#include<iostream>

#include<bits/stdc++.h>

using namespace std;

class Solution {

public:

int longestRepeatingSubstring(string S) {

int n = S.size();

S = " " + S;

int ret = 0;

vector < vector <int> > dp(n + 1, vector <int> (n + 1));

for(int i = 1; i <= n; i++){

for(int j = i + 1; j <= n; j++){

if(S[i] == S[j]){

dp[i][j] = max(dp[i][j], 1 + dp[i - 1][j - 1]);

ret = max(ret, dp[i][j]);

}

}

}

return ret;

}

};

main(){

Solution ob;

cout << (ob.longestRepeatingSubstring("abbaba"));

}

**C++ Program to find all the permutations of a string**

#include <iostream>

#include <string.h>

using namespace std;

void permute(string& a, int l, int r)

{

if (l == r)

cout << a << endl;

else {

for (int i = l; i <= r; i++) {

swap(a[l], a[i]);

permute(a, l + 1, r);

swap(a[l], a[i]);

}

}

}

int main()

{

string str;

cout<<"Enter the string: ";

cin>>str;

int n = str.size();

permute(str, 0, n - 1);

return 0;

}

1. C++ Program to determine whether two strings are the anagram

#include<iostream> #include<string.h> using namespace std; int main()

{

int temp=0;

char str1[80],str[80]; cout<<"Enter some words:\n"; cin.getline(str,80); cin.getline(str1,80);

int size=strlen(str); int size1=strlen(str1);

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

{

for(int j=0;j<size1;j++)

{

if(str[i]==str1[j]) temp++;

}

}

if(temp!=0)

cout<<"Yes too words are anagram"; return 0;

}

1. C++ Program to divide a string in 'N' equal parts. #include<iostream>

#include<string.h> #include<string> using namespace std; int main()

{

char s1[100]; cin>>s1;

int size=strlen(s1); int n;

cin>>n;

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

{if(i==n-1)

cout<<"\n"; else cout<<s1[i];

}

}

1. C++ Program to find all subsets of a string

#include <stdio.h> #include <string.h> #include<iostream> #include<string> using namespace std;

void substring(char s[], char sub[], int p, int len){ int c = 0;

while (c < len) { sub[c] = s[p+c]; c++;

}

sub[c] = '\0';

}

int main()

{

char c[10],str[100]; cin.getline(str,100);

int i, j, len = strlen(str);

cout<<"All subsets for the given string are: ";

for(i = 0; i < len; i++){

for(j = 1; j <= len-i; j++){ substring(str,c,i,j); cout<<c<<"\n";

}

}

return 0;

}

1. C++ Program to find the longest repeating sequence in a string

#include<iostream> #include<string.h> using namespace std; int main()

{

char s1[100]; cin.getline(s1,100); int size=strlen(s1); int tmax=1,i,j; for(i=0;i<size;i++)

{

for(j=i+1;j<size;j++)

{

if(s1[i]==s1[j]||s1[i]+32==s1[j])

{

tmax++; s1[j]='\0';

}

}

if (s1[i]!='\0')

cout<<s1[i]<<"-"<<tmax<<endl; tmax=1;

}

}

1. C++ Program to find all the permutations of a string #include <iostream>

#include <string> #include <algorithm>

using namespace std;

void permute (string temp\_str, int start, int end)

{

int i;

if (start == end){

cout << temp\_str << " ";

}

else{

for (int i = start; i < temp\_str.length (); ++i){ swap (temp\_str[start], temp\_str[i]); permute (temp\_str, start + 1, end);

swap (temp\_str[start], temp\_str[i]);

}

}

}

int main()

{

string input\_str; bool flag = false;

cout << "Enter String : "; cin >> input\_str;

for (int i = 0; i < input\_str.length () - 1; ++i)

{

if (input\_str[i] == input\_str[i + 1])

{

flag = true; break;

}

else {

flag = false; break;

}

}

if (flag)

{

cout << "The permutation of " << input\_str << " is : " << input\_str << endl;

}

else

{

cout << "The permutations of " << input\_str << " are : " << endl; permute (input\_str, 0, input\_str.length () - 1);

}

cout << endl; return 0;

}

1. C++ Program to remove all the white spaces from a string #include<iostream>

#include<string.h> using namespace std; int main()

{

int temp=0; char str[80];

cout<<"Enter some words:\n"; cin.getline(str,80);

int size=strlen(str); for(int i=0;i<size;i++)

{

if(str[i]==' ') continue; else cout<<str[i];

}

}

1. C++ Program to replace lower-case characters with upper-case and vice-versa

//LowerCase #include<iostream> #include<string.h> #include<string> using namespace std; int main()

{

char s1[100]; cin>>s1;

int size=strlen(s1);

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

{

if(s1[i]<97)

{

s1[i]=s1[i]+32;

}

}

cout<<s1;

}

//UpperCase

#include<iostream> #include<string.h>

using namespace std; int main()

{

char s1[100]; cin>>s1;

int size=strlen(s1);

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

{

if(s1[i]>=97)

{

s1[i]=s1[i]-32;

}

}

cout<<s1;

}

1. C++ Program to replace the spaces of a string with a specific character #include<iostream>

#include<string.h> using namespace std; int main()

{

int temp=0; char str[80];

cout<<"Enter some words:\n"; cin.getline(str,80);

int size=strlen(str); for(int i=0;i<size;i++)

{

if(str[i]==' ')

cout<<"@"; else cout<<str[i];

}

}

1. C++ Program to determine whether a given string is palindrome #include<iostream>

#include<string.h> using namespace std; int main()

{

int temp=0;

char str1[80],str[80]; cout<<"Enter some words:\n"; cin.getline(str,80);

int t=0,size=strlen(str); for(int i=size-1;i>=0;i++)

{

str1[t]=str[i]; t++;

}

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

{

for(int j=0;j<size;j++)

{

if(str[i]==str1[j]) temp++;

else temp=0;

}

}

if(temp!=0)

cout<<"Yes its a palindrom"; else

cout<<"not a palindrom";

return 0;

}

1. C++ Program to determine whether one string is a rotation of another #include<iostream>

#include<string.h> using namespace std; int main()

{

int temp=0;

char str1[80],str[80]; cout<<"Enter some words:\n";

cin.getline(str,80); cin.getline(str1,80); int size=strlen(str); int size1=strlen(str1);

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

{

for(int j=size-1;j>=0;j--)

{

if(str[i]==str1[j]) temp++;

else

{

temp=0;

break;

}

}

}

if(temp!=0)

cout<<"Yes too words are rotation of one another"; else

cout<<"Not a rotation word"; return 0;

}

1. C++ Program to find maximum and minimum occurring character in a string

#include <stdio.h> #include <string.h> #include <string> #include<iostream> #include<cstring>

using namespace std; #define ASCII\_SIZE 256 using namespace std;

char getMaxOccurringChar(char\* str)

{

int count[ASCII\_SIZE] = { 0 }; int len = strlen(str);

int max = 0; char result;

for (int i = 0; i < len; i++) { count[str[i]]++;

if (max < count[str[i]]) {

max = count[str[i]]; result = str[i];

}

}

return result;

}

int main()

{

char str[100]; cin.getline(str,100);

cout << "Max occurring character is "<< getMaxOccurringChar(str);

}

1. C++ Program to find Reverse of the string #include<iostream>

#include<string> using namespace std; int main()

{

string s; getline(cin,s);

int size=s.length();

for(int i=size;i>=0;i--)

{

cout<<s[i]; //printing the output

}

return 0;

}

1. C++ program to find the duplicate characters in a string #include<iostream>

#include<string.h> using namespace std; int main()

{

char s1[100]; cin.getline(s1,100); int size=strlen(s1); int tmax=1,max=0,i,j; char ch; for(i=0;i<size;i++)

{

for(j=i+1;j<size;j++)

{

if(s1[i]==s1[j])

{

tmax++; } if(i==0)

{

max=tmax;

}

}

if(tmax>max)

{

max=tmax; ch=s1[i];

}

tmax=1;

}

cout<<ch<<"="<<max<<endl;

}

1. C++ program to find the duplicate words in a string #include <stdio.h>

#include <string.h> #include <string> #include<iostream> #include<cstring>

using namespace std; int main()

{

char string[] = "big black bug bit a big black dog on his big black nose"; char words[100][100];

int i = 0, j = 0, k, length, count;

//Split the string into words such that each row of array words represents a word for(k=0; string[k]!='\0'; k++){

//Here, i represents row and j represents column of two-dimensional array words if(string[k] != ' ' && string[k] != '\0'){

//Converts the string into lowercase and add it to array words words[i][j++] = tolower(string[k]);

}

else{

words[i][j] = '\0';

//Increment row count to store new word i++;

//Set column count to 0 j = 0;

}

}

//Store row count in variable length length = i+1;

cout<<"Duplicate words in the given string: \n"; for(i = 0; i < length; i++){

count = 1;

for(j = i+1; j < length; j++){

if(strcmp(words[i], words[j]) == 0 && (strcmp(words[j],"0") != 0)){ count++;

//Set words[j] to 0 to avoid printing visited word strcpy(words[j],"0");

}

}

//Displays the duplicate word if count is greater than 1 if(count > 1 )

cout<<"\n"<<words[i];

}

return 0;

}

1. C++ Program to find the frequency of characters #include<iostream>

#include<string.h> using namespace std; int main()

{

char s1[100]; cin.getline(s1,100); int size=strlen(s1); int tmax=1,i,j; for(i=0;i<size;i++)

{

for(j=i+1;j<size;j++)

{

if(s1[i]==s1[j]||s1[i]+32==s1[j])

{

tmax++; s1[j]='\0';

}

}

if (s1[i]!='\0')

cout<<s1[i]<<"-"<<tmax<<endl;

tmax=1;

}

}

1. C++ Program to find the largest and smallest word in a string #include<iostream>

#include<cstring> using namespace std;

void minMaxLengthWords(string input, string &minWord, string &maxWord)

{

// minWord and maxWord are received by reference

// and not by value

// will be used to store and return output int len = input.length();

int si = 0, ei = 0;

int min\_length = len, min\_start\_index = 0, max\_length = 0, max\_start\_index = 0;

// Loop while input string is not empty while (ei <= len)

{

if (ei < len && input[ei] != ' ') ei++;

else

{

// end of a word

// find curr word length int curr\_length = ei - si;

if (curr\_length < min\_length)

{

min\_length = curr\_length; min\_start\_index = si;

}

if (curr\_length > max\_length)

{

max\_length = curr\_length; max\_start\_index = si;

}

ei++; si = ei;

}

}

// store minimum and maximum length words

minWord = input.substr(min\_start\_index, min\_length); maxWord = input.substr(max\_start\_index, max\_length);

}

// Driver code int main()

{

char a[100]; cin.getline(a,100);

string minWord, maxWord; minMaxLengthWords(a, minWord, maxWord);

// to take input in string use getline(cin, a); cout << "Minimum length word: "

<< minWord << endl

<< "Maximum length word: "

<< maxWord << endl;

}

1. C++ Program to find the most repeated word in a text file

#include <iostream> #include <cstdio> #include <cstdlib> #include <cstring>

#include <cctype>

# define MAX\_CHARS 26

# define MAX\_WORD\_SIZE 30 using namespace std;

struct TrieNode

{

bool isEnd; unsigned frequency; int indexMinHeap;

TrieNode\* child[MAX\_CHARS];

};

struct MinHeapNode

{

TrieNode\* root; unsigned frequency; char\* word;

};

struct MinHeap

{

unsigned capacity; int count;

MinHeapNode\* array;

};

TrieNode\* newTrieNode()

{

TrieNode\* trieNode = new TrieNode; trieNode->isEnd = 0;

trieNode->frequency = 0;

trieNode->indexMinHeap = -1;

for (int i = 0; i < MAX\_CHARS; ++i) trieNode->child[i] = NULL;

return trieNode;

}

MinHeap\* createMinHeap(int capacity)

{

MinHeap\* minHeap = new MinHeap; minHeap->capacity = capacity; minHeap->count = 0;

minHeap->array = new MinHeapNode [minHeap->capacity]; return minHeap;

}

void swapMinHeapNodes (MinHeapNode\* a, MinHeapNode\* b)

{

MinHeapNode temp = \*a;

\*a = \*b;

\*b = temp;

}

/\*

\* min heapify

\*/

void minHeapify(MinHeap\* minHeap, int idx)

{

int left, right, smallest; left = 2 \* idx + 1;

right = 2 \* idx + 2; smallest = idx;

if (left < minHeap->count && minHeap->array[left].frequency < minHeap-

>array[smallest].frequency) smallest = left;

if (right < minHeap->count && minHeap->array[right].frequency < minHeap-

>array[smallest].frequency) smallest = right;

if (smallest != idx)

{

minHeap->array[smallest].root->indexMinHeap = idx; minHeap->array[idx].root->indexMinHeap = smallest;

swapMinHeapNodes (&minHeap->array[smallest], &minHeap->array[idx]); minHeapify(minHeap, smallest);

}

}

/\*

\* build a heap

\*/

void buildMinHeap(MinHeap\* minHeap)

{

int n, i;

n = minHeap->count - 1;

for (i = ( n - 1 ) / 2; i >= 0; --i) minHeapify(minHeap, i);

}

/\*

\* inserts a word to heap

\*/

void insertInMinHeap(MinHeap\* minHeap, TrieNode\*\* root, const char\* word)

{

if ((\*root)->indexMinHeap != -1)

{

++( minHeap->array[(\*root)->indexMinHeap].frequency); minHeapify(minHeap, (\*root)->indexMinHeap);

}

else if (minHeap->count < minHeap->capacity)

{

int count = minHeap->count;

minHeap->array[count].frequency = (\*root)->frequency;

minHeap->array[count].word = new char [strlen( word ) + 1]; strcpy(minHeap->array[count].word, word);

minHeap->array[count].root = \*root;

(\*root)->indexMinHeap = minHeap->count;

++( minHeap->count ); buildMinHeap( minHeap );

}

else if ((\*root)->frequency > minHeap->array[0].frequency)

{

minHeap->array[0].root->indexMinHeap = -1; minHeap->array[0].root = \*root;

minHeap->array[0].root->indexMinHeap = 0; minHeap->array[0].frequency = (\*root)->frequency; delete [] minHeap->array[0].word;

minHeap->array[0]. word = new char [strlen( word ) + 1]; strcpy( minHeap->array[0].word, word );

minHeapify (minHeap, 0);

}

}

/\*

\* Inserts a new word to both Trie and Heap

\*/

void insertUtil (TrieNode\*\* root, MinHeap\* minHeap, const char\* word, const char\* dupWord)

{

if (\*root == NULL)

\*root = newTrieNode(); if (\*word != '\0')

insertUtil (&((\*root)->child[tolower( \*word ) - 97]), minHeap, word + 1, dupWord); else

{

if ((\*root)->isEnd)

++((\*root)->frequency); else

{

(\*root)->isEnd = 1;

(\*root)->frequency = 1;

}

insertInMinHeap(minHeap, root, dupWord);

}

}

/\*

\* add a word to Trie & min heap

\*/

void insertTrieAndHeap(const char \*word, TrieNode\*\* root, MinHeap\* minHeap)

{

insertUtil(root, minHeap, word, word);

}

/\*

\* Display Min Heap

\*/

void displayMinHeap(MinHeap\* minHeap)

{

int i;

for (i = 0; i < minHeap->count; ++i)

{

cout<<minHeap->array[i].word<<" : "<<minHeap->array[i].frequency<<endl;

}

}

/\*

\* takes a file as input, add words to heap and Trie

\*/

void printKMostFreq(FILE\* fp, int k)

{

MinHeap\* minHeap = createMinHeap(k); TrieNode\* root = NULL;

char buffer[MAX\_WORD\_SIZE]; while (fscanf(fp, "%s", buffer) != EOF)

insertTrieAndHeap(buffer, &root, minHeap); displayMinHeap(minHeap);

}

/\*

\* Main

\*/

int main()

{

int k = 5;

FILE \*fp = fopen("file.txt", "r"); if (fp == NULL)

printf ("File doesn't exist "); else

printKMostFreq (fp, k); return 0;

}

1. C++ Program to find the number of the words in the given text file #include <iostream>

#include <fstream> using namespace std; int main()

{

ifstream fin("fname.txt"); int word=1;

char ch;

fin.seekg(0,ios::beg);

while(fin)

{

fin.get(ch); if(ch==' '||ch=='\n') word++;

}

cout<<"\nWords="<<word<<"\n"; fin.close();

return 0;

}

1. C++ Program to separate the Individual Characters from a String #include<iostream>

#include<string.h> using namespace std; int main()

{

char s1[100]; cin.getline(s1,100);

int size=strlen(s1); int tmax=1,i,j; for(i=0;i<size;i++)

{

for(j=i+1;j<size;j++)

{

if(s1[i]==s1[j]||s1[i]+32==s1[j])

{

tmax++; s1[j]='\0';

}

}

if (s1[i]!='\0')

cout<<s1[i]<<"-"<<tmax<<endl; tmax=1;

}

}

1. C++ Program to swap two string variables without using third or temp variable. #include<iostream>

#include<string.h> #include<string> #include<cstring> using namespace std;

int main()

{

char s1[100],s2[100]; cin.getline(s1,100); cin.getline(s2,100); cout<<"original :"<<endl; cout<<"s1="<<s1<<endl; cout<<"s2="<<s2<<endl; int size1=strlen(s1);

int t=0,t1=0,i,j; strcat(s1,s2);

int size2=strlen(s1); for(i=0;i<size1;i++)

{

s2[i]=s1[i];

}

for(i=0;i<size2;i++)

{

if(s2[i]==s1[i])

s1[i]='\0';

else

{

s1[t]=s1[i]; t++;}

}

for(i=size1;i<size2;i++)

{ s1[i]='\0';

}

cout<<"After Swap :"<<endl; cout<<"s1="<<s1<<endl; cout<<"s2="<<s2<<endl;

}

1. C++ Program to print smallest and biggest possible palindrome word in a given string

#include <stdio.h> #include <stdbool.h> #include <string.h> #include<iostream> using namespace std;

bool isPalindrome(char a[]){ bool flag = true;

for(int i = 0; i < strlen(a)/2; i++){ if(a[i] != a[strlen(a)-i-1]){

flag = false; break;

}

}

return flag;

}

int main()

{

char str[100] ; cin.getline(str,100);

char words[100][100], smallPalin[100], bigPalin[100]; int i = 0, j = 0, k, length, count = 0;

for(k=0; str[k]!='\0'; k++)

if(str[k] != ' ' && str[k] != '\0'){ words[i][j++] = tolower(str[k]);

}

else{

words[i][j] = '\0';

//Increment row count to store new word i++;

//Set column count to 0 j = 0;

}

}

//Store row count in variable length length = i+1;

//Determine the smallest and biggest palindromes in a given string for(int i = 0; i < length; i++){

if(isPalindrome(words[i])){

count++;

//When first palindromic word is found if(count == 1){

//Initialize smallPalin and bigPalin with first palindromic word strcpy(smallPalin, words[i]);

strcpy(bigPalin, words[i]);

}

//Compare smallPalin and bigPalin with each palindromic words else{

//If length of smallPalin is greater than next palindromic word

//Store that word in smallPalin if(strlen(smallPalin) > strlen(words[i]))

strcpy(smallPalin, words[i]);

//If length of bigPalin is less than next palindromic word

//Store that word in bigPalin if(strlen(bigPalin) < strlen(words[i]))

strcpy(bigPalin, words[i]);

}

}

}

if(count == 0)

cout<<"No palindrome is present in the given string"; else{

cout<<"Smallest palindromic word: "<<smallPalin<<endl; cout<<"Biggest palindromic word: "<<bigPalin;

}

return 0;

}