**DR B R AMBEDKAR NATIONAL INSTITUE OF TECHNOLOGY JALANDHAR**

****

Department: Computer Science and Engineering

LAB FILE OF SYSTEM PROGRAMMING

(CSX-326)

SESSION: JAN - MAY 2020

**SUBMITTED TO: SUBMITTED BY:**

Dr. Mohit Kumar Ankit Goyal

Assistant Professor Roll No : 17103011

I.T. Department Group : G1

Branch : CSE

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S No.** | **Program** | **Page** | **Remark** |
| 1. | WAP for Linear Search | 3 |  |
| 2. | WAP for Binary Search | 4-5 |  |
| 3. | WAP for Merge Sorting | 6-7 |  |
| 4. | WAP for Counting Sorting | 8-9 |  |
| 5. | WAP for Radix Sorting | 10-11 |  |
| 6. | WAP for Bucket Sorting | 12-13 |  |
| 7. | WAP for Traveling Salesman Problem | 14- 15 |  |
| 8. | Write a Lex program to count number of tokens in a file | 16 |  |
| 9. | Write a Lex program for calculator | 17 |  |
| 10. | Write a lex and yacc program to check if a string is palindrome | 18-20 |  |
| 11. | WAP for Huffman Coding | 21-22 |  |
| 12. | WAP for to implement top down parser | 23-26 |  |
| 13. | WAP for to implement bottom up parsing | 27-29 |  |
| 14. | Design and Implementation of editor in any language | 30-35 |  |
| 15. | Design and implement one pass Assembler in any language | 36-41 |  |
| 16. | Design and implement two pass assembler in any language | 42-46 |  |
| 17. | Design and implement of two pass macro processor | 47-49 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Program No. 1**

**Aim:** Write a program for Linear Search.

**Description:**  linear search is a method for finding an element within a list It sequentially checks each element of the list until a match is found or the whole list has been searched. If the algorithm reaches the end of the list, the search terminates unsuccessfully. Linear search is usually very simple to implement, and is practical when the list has only a few elements, or when performing a single search in an un-ordered list.

Complexity: (i) best case-O(1) (ii) average case-O(n) (iii) worst case-O(n)

**Program:**

#include<bits/stdc++.h>

using namespace std;

main()

{

int n,x,i;

cout<<"enter no. of elements and element to be search\n ";

cin>>n>>x;

int arr[n];

cout<<"enter elements: ";

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

cin>>arr[i];

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

{

if(arr[i]==x)

{

cout<<"no. is present"<<i+1<<"\n";

break;

}

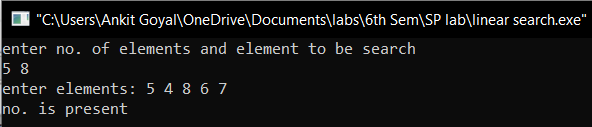
}

if(i==n)

cout<<"no. is not present\n";

}

**Output:**



**Program No. 2**

**Aim:** Write a program for Binary Search.

**Description:**  Binary Search is a [search algorithm](https://en.wikipedia.org/wiki/Search_algorithm) that finds the position of a target value within a sorted array. Binary search compares the target value to the middle element of the array. If they are not equal, the half in which the target cannot lie is eliminated and the search continues on the remaining half, again taking the middle element to compare to the target value, and repeating this until the target value is found. If the search ends with the remaining half being empty, the target is not in the array.

Complexity: (i) best case-O(1) (ii) average case-O(logn) (iii) worst case-O(logn)

**Program:**

#include<bits/stdc++.h>

using namespace std;

int main()

{

int n;

cout<<"enter no of elements\n";

cin>>n;

cout<<"enter array\n";

int arr[n];

int i=0;

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

{

cin>>arr[i];

}

sort(arr,arr+n);

int num;

cout<<"enter number to find\n";

cin>>num;

int mid,beg=0,en=n-1;

mid=(en-beg)/2;

while(num!=arr[mid]&&beg<en)

{

if(num>=arr[mid])

{

beg=mid+1;

}

else

{

en=mid-1;

}

mid=beg+(en-beg)/2;

}

if(num==arr[mid])

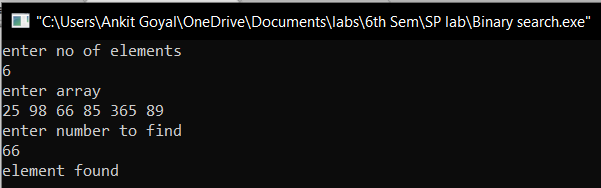
cout<<"element found\n";

else

cout<<"not found\n";

}

**Output:**



**Program No. 3**

**Aim:** Write a program for Merge Sort.

**Description:**  Merge Sort is a [Divide and Conquer](https://www.geeksforgeeks.org/divide-and-conquer-introduction/) algorithm. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves. The merge() function is used for merging two halves. The merge(arr, l, m, r) is key process that assumes that arr[l..m] and arr[m+1..r] are sorted and merges the two sorted sub-arrays into one.

Complexity: (i) best case-O(nlogn) (ii) average case-O(nlogn)(iii) worstcase-O(nlogn)

**Program:**

#include<iostream>

using namespace std;

void merge\_arrays(int arr[],int beg,int mid,int end)

{

int n1,n2,i,j;

n1=mid-beg+1;

n2=end-mid;

int L\_arr[n1+1],R\_arr[n2+1];

for(i=0;i<n1;i++)

L\_arr[i]=arr[beg+i];

L\_arr[n1]=1000000;

for(j=0;j<n2;j++)

R\_arr[j]=arr[mid+j+1];

R\_arr[n2]=1000000;

int k;

i=0;j=0;

for(k=beg;k<=end;k++)

{

if(L\_arr[i]<R\_arr[j])

{

arr[k]=L\_arr[i];

i++;

}

else

{

arr[k]=R\_arr[j];

j++;

}

}

}

void merge\_sort(int arr[],int beg,int end)

{

if(beg<end)

{

int mid;

mid=(beg+end)/2;

merge\_sort(arr,beg,mid);

merge\_sort(arr,mid+1,end);

merge\_arrays(arr,beg,mid,end);

}

}

main()

{

int n,i,j;

cout<<"enter no. of elements\n";

cin>>n;

int arr[n];

cout<<"enter elements: ";

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

cin>>arr[i];

merge\_sort(arr,0,n-1);

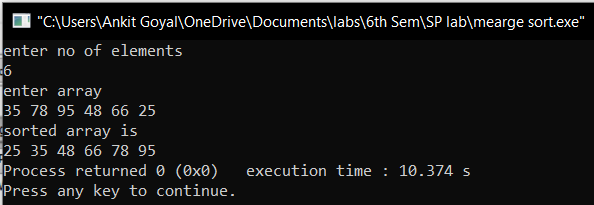
cout<<"sorted array is:\n";

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

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

}

**Output:**



**Program No. 4**

**Aim:** Write a program for Counting Sort.

**Description:**  counting sort is an [algorithm](https://en.wikipedia.org/wiki/Algorithm) for [sorting](https://en.wikipedia.org/wiki/Sorting_algorithm) a collection of objects according to keys that are small [integers](https://en.wikipedia.org/wiki/Integer); that is, it is an [integer sorting](https://en.wikipedia.org/wiki/Integer_sorting) algorithm. It operates by counting the number of objects that have each distinct key value, and using arithmetic on those counts to determine the positions of each key value in the output sequence. Its running time is linear in the number of items and the difference between the maximum and minimum key values, so it is only suitable for direct use in situations where the variation in keys is not significantly greater than the number of items.

Complexity: (i) best case-O(n+k) (ii) average case-O(n+k) (iii) worst case-O(n+k)

**Program:**

#include<bits/stdc++.h>

using namespace std;

void counting\_sort(int arr[],int sorted\_arr[], int n, int range)

{

int minm=(\*min\_element(arr,arr+n));

int temp\_arr[range]={0};

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

temp\_arr[arr[j]-minm]++;

for(int i=1;i<range;i++)

temp\_arr[i]=temp\_arr[i]+temp\_arr[i-1];

for(int i=n-1;i>=0;i--)

{

sorted\_arr[temp\_arr[arr[i]-minm]-1]=arr[i];

temp\_arr[arr[i]-minm]--;

}

}

main()

{

int n,range;

cout<<"enter no. of elements\n";

cin>>n;

int a[n],sorted\_arr[n],i;

cout<<"enter elements: ";

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

cin>>a[i];

range = (\*max\_element(a,a+n)) - (\*min\_element(a,a+n)) + 1;

counting\_sort(a,sorted\_arr,n,range);

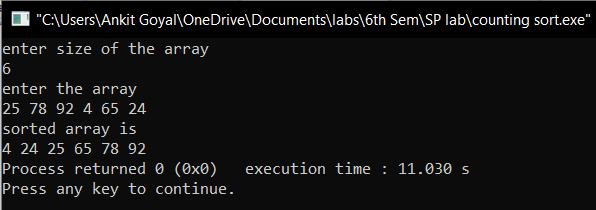
cout<<"sorted elements:\n";

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

cout<<sorted\_arr[i]<<" ";

}

**Output:**



**Program No. 5**

**Aim:** Write a program for Radix Sort.

**Description:**  Radix sort is a non-[comparative](https://en.wikipedia.org/wiki/Comparison_sort) [sorting algorithm](https://en.wikipedia.org/wiki/Sorting_algorithm). It avoids comparison by creating and [distributing](https://en.wikipedia.org/wiki/Distribution_sort) elements into buckets according to their [radix](https://en.wikipedia.org/wiki/Radix). For elements with more than one [significant digit](https://en.wikipedia.org/wiki/Significant_digit), this bucketing process is repeated for each digit, while preserving the ordering of the prior step, until all digits have been considered. For this reason, radix sort has also been called [bucket sort](https://en.wikipedia.org/wiki/Bucket_sort) and digital sort.

Complexity: (i) best case-O(nk) (ii) average case-O(nk) (iii) worst case-O(nk)

**Program:**

#include<bits/stdc++.h>

using namespace std;

void countSort(int arr[], int n, int exp)

{

int output[n];

int i,temp\_arr[10]={0};

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

temp\_arr[(arr[i]/exp)%10]++;

for(i=1;i<10;i++)

temp\_arr[i]+=temp\_arr[i-1];

for(i=n-1;i>=0;i--)

{

output[temp\_arr[(arr[i]/exp)%10]-1]=arr[i];

temp\_arr[(arr[i]/exp)%10]--;

}

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

arr[i]=output[i];

}

void print(int arr[], int n)

{

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

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

cout<<"\n";

}

void radixsort(int arr[], int n)

{

int maxm = \*max\_element(arr,arr+n);

for (int exp=1;(maxm/exp)>0;exp\*=10)

{

countSort(arr, n, exp);

cout<<"\n";

cout<<"sorting on the basis of "<<exp<<"th digit\n";

print(arr,n);

}

}

int main()

{

int n;

cout<<"enter no. of elements: ";

cin>>n;

cout<<"enter elements: ";

int arr[n];

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

cin>>arr[i];

radixsort(arr, n);

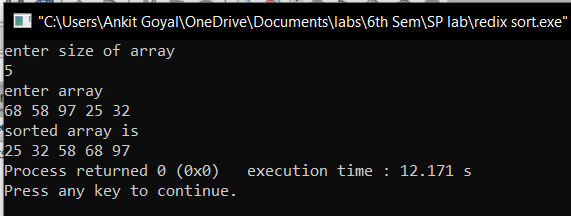
cout<<"sorted array is: ";

print(arr,n);

cout<<"\n";

}

**Output:**



**Program No. 6**

**Aim:** Write a program for Bucket Sort.

**Description:**  Bucket sort is a [sorting algorithm](https://en.wikipedia.org/wiki/Sorting_algorithm) that works by distributing the elements of an [array](https://en.wikipedia.org/wiki/Array_data_structure) into a number of [buckets](https://en.wikipedia.org/wiki/Bucket_(computing)). Each bucket is then sorted individually, either using a different sorting algorithm, or by recursively applying the bucket sorting algorithm. It is a [distribution sort](https://en.wikipedia.org/wiki/Distribution_sort), a generalization of [pigeonhole sort](https://en.wikipedia.org/wiki/Pigeonhole_sort), and is a cousin of [radix sort](https://en.wikipedia.org/wiki/Radix_sort) in the most-to-least significant digit flavor. Bucket sort can be implemented with comparisons and therefore can also be considered a [comparison sort](https://en.wikipedia.org/wiki/Comparison_sort) algorithm. The [computational complexity](https://en.wikipedia.org/wiki/Analysis_of_algorithms) depends on the algorithm used to sort each bucket, the number of buckets to use, and whether the input is uniformly distributed.

Complexity: (i) best case-O(n+k) (ii) average case-O(n+k) (iii) worst case-O(n2)

**Program:**

#include<bits/stdc++.h>

using namespace std;

void bucketSort(float arr[], int n)

{

vector<float> b[n];

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

{

int bi = n\*arr[i];

b[bi].push\_back(arr[i]);

}

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

sort(b[i].begin(), b[i].end());

int index = 0;

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

for (int j = 0; j < b[i].size(); j++)

arr[index++] = b[i][j];

}

void printArray(float arr[] , int n, int div)

{

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

cout << arr[i]\*div << " ";

cout<<endl;

}

int main()

{

int n;

cout<<"enter the number of elements in the array\n";

cin>>n;

float arr[n];

cout<<"enter the elements\n";

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

cin>>arr[i];

int max1 = \*max\_element(arr,arr+n);

int div = 1;

while(max1>0)

{

div= div\*10;

max1=max1/10;

}

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

arr[i]=arr[i]/div;

bucketSort(arr, n);

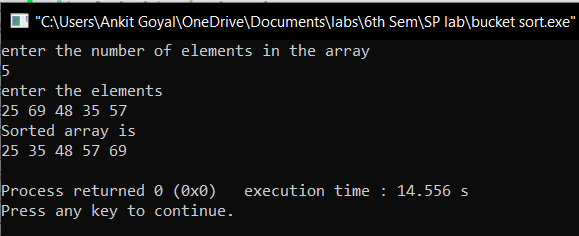
cout << "Sorted array is \n";

printArray(arr , n, div);

return 0;

}

**Output:**



**Program No. 7**

**Aim:** Write a program for Traveling Salesman Problem.

**Description:** Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point.

**Program:**

#include <bits/stdc++.h>

using namespace std;

int travllingSalesmanProblem(int\*\* graph,int s,int n)

{

vector<int> vertex;

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

{

if(i!=s)

{

vertex.push\_back(i);

}

}

int min\_path=INT\_MAX;

do {

int current\_pathweight=0;

int k=s;

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

{

current\_pathweight += graph[k][vertex[i]];

k=vertex[i];

}

current\_pathweight += graph[k][s];

min\_path = min(min\_path,current\_pathweight);

}

while(next\_permutation(vertex.begin(),vertex.end()));

return min\_path;

}

int main()

{

int n,i,j;

cout<<"enter number of cities\n";

cin>>n;

cout<<"enter the distances\n";

int \*\*matrix=new int\*[n];

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

{

matrix[i]=new int[n];

}

int v=0;

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

{

for(j=v;j<n;j++)

{

if(i==j)

{

matrix[i][j]=0;

}

else

{

cin>>matrix[i][j];

matrix[j][i]=matrix[i][j];

}

}

v++;

}

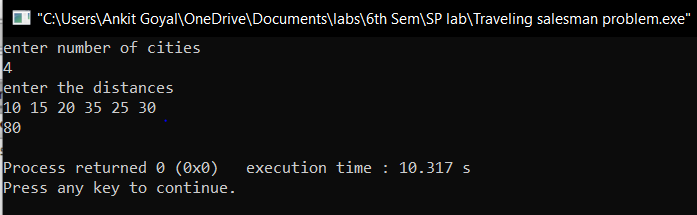
int s = 0;

cout << travllingSalesmanProblem(matrix,s,n)<<"\n";

return 0;

}

**Output:**



**Program No. 8**

**Aim:** Write a Lex program to count number of tokens in a file.

**Description:** Lex is a computer program that generates lexical analyzers and was written by Mike Lesk and Eric Schmidt.

Lex reads an input stream specifying the lexical analyzer and outputs source code implementing the lexer in the C programming language. This program counts the number of tokens.

**Program:**

%{

#include<stdio.h>

#include<string.h>

int i = 0;

%}

/\* Rules Section\*/

%%

/\* Rule for counting number of tokens\*/

"while"|"if"|"else" {i++;}

"int"|"float" {i++;}

[a-zA-Z\_][a-zA-Z0-9\_]\* {i++;}

"<="|"=="|"="|"++"|"-"|"\*"|"+"|"<"|">" {i++;}

[(){}|, ;] {i++;}

[0-9]\*"."[0-9]+ {i++;}

[0-9]+ {i++;}

"\n" {printf("no of tokens : %d\n", i); i = 0;}

%%

int yywrap(void){}

int main()

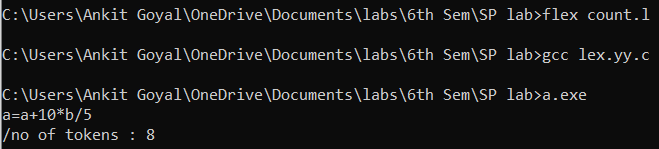
{

// The function that starts the analysis

yylex();

}

**Output:**



**Program No. 9**

**Aim:** Write a Lex program for calculator .

**Description:** Lex reads an input stream specifying the lexical analyzer and outputs source code implementing the lexer in the C programming language. This program counts number of vowels and consonants in the input string.

**Program:**

%{

int vow\_count=0;

int const\_count =0;

%}

%%

[aeiouAEIOU] {vow\_count++;}

[a-zA-Z] {const\_count++;}

%%

int yywrap(){}

int main()

{

printf("Enter the string of vowels and consonents:");

yylex();

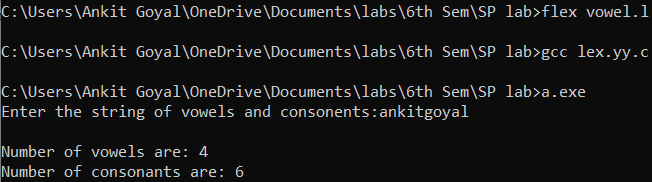
printf("Number of vowels are: %d\n", vow\_count);

printf("Number of consonants are: %d\n", const\_count);

return 0;

}

**Output:**



**Program No. 10**

**Aim:** Write a lex and yacc program to check if a string is palindrome.

**Description:** A parser generator is a program that takes as input a specification of a syntax, and produces as output a procedure for recognizing that language. Historically, they are also called compiler-compilers. Yet another compiler-compiler is an LALR(1) parser generator. YACC was originally designed for being complemented by Lex.

**Program:**

**Lex File:**

%{

/\* Definition section \*/

#include <stdio.h>

#include <stdlib.h>

#include "y.tab.h"

%}

/\* %option noyywrap \*/

/\* Rule Section \*/

%%

[a-zA-Z]+ {yylval.f = yytext; return STR;}

[-+()\*/] {return yytext[0];}

[ \t\n] {;}

%%

int yywrap()

{

return -1;

}

**YACC File:**

%{

/\* Definition section \*/

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

extern int yylex();

void yyerror(char \*msg);

int flag,i,k=0;

%}

%union {

char\* f;

}

%token <f> STR

%type <f> E

/\* Rule Section \*/

%%

S : E {

flag = 0;

k = strlen($1) - 1;

if(k%2==0){

for (i = 0; i <= k/2; i++)

{

if ($1[i] == $1[k-i]) {

} else

flag = 1;

}

if (flag == 1) printf("Not palindrome\n");

else printf("palindrome\n");

printf("%s\n", $1);

}else{

for (i = 0; i < k/2; i++) {

if ($1[i] == $1[k-i]) {

} else {

flag = 1;

}

}

if (flag == 1) printf("Not palindrome\n");

else printf("palindrome\n");

printf("%s\n", $1);

}

};

E : STR {$$ = $1;};

%%

void yyerror(char \*msg)

{

fprintf(stderr, "%s\n", msg);

exit(1);

}

int main()

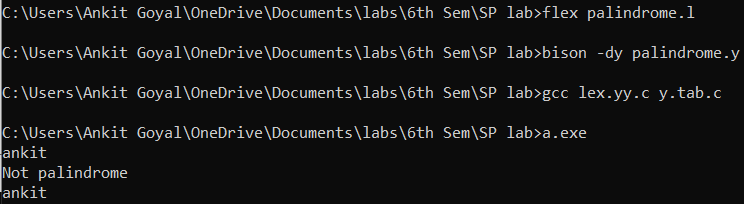
{

yyparse();

return 0;

}

**Output:**



**Program No. 11**

**Aim:** Write a program for Huffman Coding.

**Description:** Huffman code is a particular type of optimal prefix code that is commonly used for lossless data compression. The output from Huffman's algorithm can be viewed as a [variable-length code](https://en.wikipedia.org/wiki/Variable-length_code) table for encoding a source symbol (such as a character in a file). The algorithm derives this table from the estimated probability or frequency of occurrence (weight) for each possible value of the source symbol.

**Program:**

#include <bits/stdc++.h>

using namespace std;

struct MinHeapNode

{

char data;

unsigned freq;

MinHeapNode \*left, \*right;

MinHeapNode(char data, unsigned freq)

{

left = right = NULL;

this->data = data;

this->freq = freq;

}

};

struct compare

{

bool operator()(MinHeapNode\* l, MinHeapNode\* r)

{

return (l->freq > r->freq);

}

};

void printCodes(struct MinHeapNode\* root, string str)

{

if (!root)

return;

if (root->data != '$')

cout << root->data << ": " << str << "\n";

printCodes(root->left, str + "0");

printCodes(root->right, str + "1");

}

void HuffmanCodes(char data[], int freq[], int size)

{

struct MinHeapNode \*left,\*right,\*top;

priority\_queue<MinHeapNode\*, vector<MinHeapNode\*>, compare> minHeap;

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

{

minHeap.push(new MinHeapNode(data[i], freq[i]));

}

while (minHeap.size() != 1)

{

left = minHeap.top();

minHeap.pop();

right = minHeap.top();

minHeap.pop();

top = new MinHeapNode('$', left->freq + right->freq);

top->left = left;

top->right = right;

minHeap.push(top);

}

printCodes(minHeap.top(), "");

}

int main()

{

int size;

cout<<"enter the no of characters\n";

cin>>size;

char arr[size];

int freq[size];

cout<<"enter the characters with frequency\n";

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

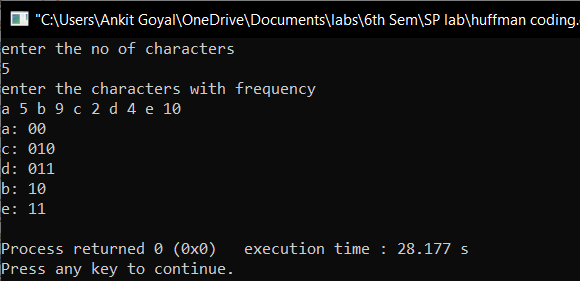
cin>>arr[i]>>freq[i];

HuffmanCodes(arr, freq, size);

return 0;

}

**Output:**



**Practical-12**

**Aim: Write a program to implement top down parser.**

**Program:**

#include<iostream>

#include<conio.h>

#include<string.h>

#include<bits/stdc++.h>

using namespace std;

class parse

{

int nt,t,m[20][20],i,s,n,p1,q,k,j;

char p[30][30],n1[20],t1[20],ch,b,c,f[30][30],fl[30][30];

public:

int scant(char);

int scannt(char);

void process();

void input();

};

int parse::scannt(char a)

{

int c=-1,i;

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

{

if(n1[i]==a)

{

return i;

}

}

return c;

}

int parse::scant(char b)

{

int c1=-1,j;

for(j=0;j<t;j++)

{

if(t1[j]==b)

{

return j;

}

}

return c1;

}

void parse::input()

{

cout<<"Enter the number of productions:";

cin>>n;

cout<<"Enter the productions one by one"<<endl;

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

cin>>p[i];

nt=0;

t=0;

}

void parse::process()

{

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

{

if(scannt(p[i][0])==-1)

n1[nt++]=p[i][0];

}

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

{

for(j=3;j<strlen(p[i]);j++)

{

if(p[i][j]!='e')

{

if(scannt(p[i][j])==-1)

{

if((scant(p[i][j]))==-1)

t1[t++]=p[i][j];

}

}

}

}

t1[t++]='$';

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

{

for(j=0;j<t;j++)

m[i][j]=-1;

}

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

{

cout<<"Enter first["<<n1[i]<<"]:";

cin>>f[i];

}

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

{

cout<<"Enter follow["<<n1[i]<<"]:";

cin>>fl[i];

}

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

{

p1=scannt(p[i][0]);

if((q=scant(p[i][3]))!=-1)

m[p1][q]=i;

if((q=scannt(p[i][3]))!=-1)

{

for(j=0;j<strlen(f[q]);j++)

m[p1][scant(f[q][j])]=i;

}

if(p[i][3]=='e')

{

for(j=0;j<strlen(fl[p1]);j++)

m[p1][scant(fl[p1][j])]=i;

}

}

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

cout<<"\t"<<t1[i];

cout<<endl;

for(j=0;j<nt;j++)

{

cout<<n1[j];

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

{

cout<<"\t"<<" ";

if(m[j][i]!=-1)

cout<<p[m[j][i]];

}

cout<<endl;

}

}

int main()

{

parse p;

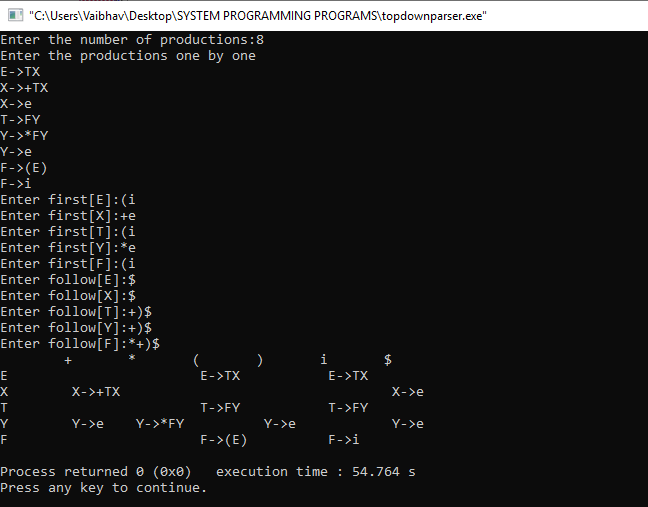
p.input();

p.process();

\_getch();

}

**Output:**



**Practical-13**

**Aim: Write a program to implement bottom up parsing**

**Program:**

#include<conio.h>

#include<iostream>

#include<string.h>

using namespace std;

struct grammer{

char p[20];

char prod[20];

}g[10];

int main()

{

int i,stpos,j,k,l,m,o,p,f,r;

int np,tspos,cr;

cout<<"\nEnter Number of productions:";

cin>>np;

char sc,ts[10];

cout<<"\nEnter productions:\n";

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

{

cin>>ts;

strncpy(g[i].p,ts,1);

strcpy(g[i].prod,&ts[3]);

}

char ip[10];

cout<<"\nEnter Input:";

cin>>ip;

int lip=strlen(ip);

char stack[10];

stpos=0;

i=0;

sc=ip[i];

stack[stpos]=sc;

i++;stpos++;

cout<<"\n\nStack\tInput\tAction";

do

{

r=1;

while(r!=0)

{

cout<<"\n";

for(p=0;p<stpos;p++)

{

cout<<stack[p];

}

cout<<"\t";

for(p=i;p<lip;p++)

{

cout<<ip[p];

}

if(r==2)

{

cout<<"\tReduced";

}

else

{

cout<<"\tShifted";

}

r=0;

\_getch();

for(k=0;k<stpos;k++)

{

f=0;

for(l=0;l<10;l++)

{

ts[l]='\0';

}

tspos=0;

for(l=k;l<stpos;l++)

{

ts[tspos]=stack[l];

tspos++;

}

for(m=0;m<np;m++)

{

cr = strcmp(ts,g[m].prod);

if(cr==0)

{

for(l=k;l<10;l++)

{

stack[l]='\0';

stpos--;

}

stpos=k;

strcat(stack,g[m].p);

stpos++;

r=2;

}

}

}

}

sc=ip[i];

stack[stpos]=sc;

i++;stpos++;

}

while(strlen(stack)!=1 && stpos!=lip);

if(strlen(stack)==1)

{

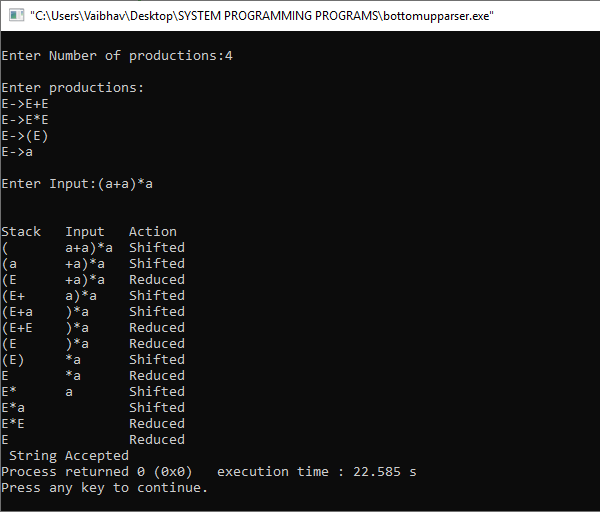
cout<<"\n String Accepted";

}

\_getch();

}

**Output:**



**Practical-14**

**Aim: Design and Implementation of editor in any language.**

**THEORY :-**

To create a simple text editor:

* First, we will create a frame ‘f’ titled “editor” and apply a metal look and feel and set an ocean theme in it.
* We will add a text area and a menubar with three menu File, Edit, and Close.
  + The “File” option has 4 menu items new, open, save and print.
  + “Edit” has 3 menu items cut, copy and paste. We will add an action listener to all the menu items(using addActionListener() function) to detect any action.
* We will add the menu items to the menu and menu to the menubar using add() function and we would add the menubar to the frame using addJMenuBar() function.
* We will add the text area to the frame using add function set the size of the frame to 500,500 using setSize(500,500) function and then display the frame using show function.

**PROGRAM:**

import java.io.\*;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class notepad extends KeyAdapter implements ActionListener, KeyListener {

static int active = 0;

static int fsize = 17;

JFrame frame1;

JMenuBar npMenuBar;

JMenu file, edit, format, view;

JMenuItem newdoc, opendoc, exit, savedoc, saveasdoc, copydoc, pastedoc, remdoc, fontfamily, fontstyle, fontsize,

status;

JTextArea maintext;

JTextField title;

Font font1;

JPanel bottom;

JLabel details, pastecopydoc;

@SuppressWarnings("rawtypes")

JList familylist, stylelist, sizelist;

JScrollPane sb;

String familyvalue[] = { "Agency FB", "Antiqua", "Architect", "Arial", "Calibri", "Comic Sans", "Courier",

"Cursive", "Impact", "Serif" };

String sizevalue[] = { "5", "10", "15", "20", "25", "30", "35", "40", "45", "50", "55", "60", "65", "70" };

int[] stylevalue = { Font.PLAIN, Font.BOLD, Font.ITALIC };

String[] stylevalues = { "PLAIN", "BOLD", "ITALIC" };

String ffamily, fsizestr, fstylestr;

int fstyle;

int cl;

int linecount;

String tle;

String topicstitle = "";

JScrollPane sp;

notepad() {

frame1 = new JFrame("Notepad Fast");

font1 = new Font("Arial", Font.PLAIN, 17);

bottom = new JPanel();

details = new JLabel();

pastecopydoc = new JLabel();

familylist = new JList(familyvalue);

stylelist = new JList(stylevalues);

sizelist = new JList(sizevalue);

familylist.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

sizelist.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

stylelist.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

bottom.add(details);

maintext = new JTextArea();

sp = new JScrollPane(maintext);

title = new JTextField(100);

sb = new JScrollPane(maintext);

maintext.setMargin(new Insets(5, 5, 5, 5));

maintext.setFont(font1);

frame1.add(maintext);

npMenuBar = new JMenuBar();

file = new JMenu("File");

edit = new JMenu("Edit");

format = new JMenu("Format");

view = new JMenu("View");

newdoc = new JMenuItem("New Document");

opendoc = new JMenuItem("Open Document");

savedoc = new JMenuItem("Save Document");

saveasdoc = new JMenuItem("Save As Document");

exit = new JMenuItem("Exit");

copydoc = new JMenuItem("Copy Document");

remdoc = new JMenuItem("Remove Document");

pastedoc = new JMenuItem("Paste Document");

fontfamily = new JMenuItem("Set Font Family");

fontstyle = new JMenuItem("Set Font Style");

fontsize = new JMenuItem("Set Font Size");

status = new JMenuItem("Status");

file.add(newdoc);

file.add(opendoc);

file.add(savedoc);

file.add(saveasdoc);

file.add(exit);

edit.add(copydoc);

edit.add(pastedoc);

edit.add(remdoc);

format.add(fontfamily);

format.add(fontstyle);

format.add(fontsize);

view.add(status);

npMenuBar.add(file);

npMenuBar.add(edit);

npMenuBar.add(format);

npMenuBar.add(view);

frame1.setJMenuBar(npMenuBar);

frame1.add(bottom, BorderLayout.SOUTH);

newdoc.addActionListener(this);

copydoc.addActionListener(this);

pastedoc.addActionListener(this);

remdoc.addActionListener(this);

status.addActionListener(this);

savedoc.addActionListener(this);

saveasdoc.addActionListener(this);

fontfamily.addActionListener(this);

fontsize.addActionListener(this);

fontstyle.addActionListener(this);

exit.addActionListener(this);

maintext.addKeyListener(this);

frame1.setSize(600, 600);

frame1.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame1.setLocationRelativeTo(null);

frame1.setVisible(true);

}

public void actionPerformed(ActionEvent ae) {

if (ae.getSource() == newdoc) {

frame1.setTitle("New Document.txt");

maintext.setText("");

title.setText("");

} else if (ae.getSource() == copydoc) {

String texts = maintext.getText();

pastecopydoc.setText(texts);

JOptionPane.showMessageDialog(null, "Copy Text " + texts);

} else if (ae.getSource() == remdoc) {

maintext.setText("");

JOptionPane.showMessageDialog(null, "Removed");

} else if (ae.getSource() == pastedoc) {

if (maintext.getText().length() != 0) {

maintext.setText(maintext.getText());

} else {

maintext.setText(pastecopydoc.getText());

}

} else if (ae.getSource() == status) {

try {

if (active == 0) {

File f = new File(tle + ".txt");

details.setText("Size: " + f.length());

}

} catch (Exception e) {

}

} else if (ae.getSource() == fontfamily) {

JOptionPane.showConfirmDialog(null, familylist, "Choose Font Family", JOptionPane.OK\_CANCEL\_OPTION,

JOptionPane.PLAIN\_MESSAGE);

ffamily = String.valueOf(familylist.getSelectedValue());

font1 = new Font(ffamily, fstyle, fsize);

maintext.setFont(font1);

} else if (ae.getSource() == fontstyle) {

JOptionPane.showConfirmDialog(null, stylelist, "Choose Font Style", JOptionPane.OK\_CANCEL\_OPTION,

JOptionPane.PLAIN\_MESSAGE);

fstyle = stylevalue[stylelist.getSelectedIndex()];

font1 = new Font(ffamily, fstyle, fsize);

maintext.setFont(font1);

} else if (ae.getSource() == fontsize) {

JOptionPane.showConfirmDialog(null, sizelist, "Choose Font Size", JOptionPane.OK\_CANCEL\_OPTION,

JOptionPane.PLAIN\_MESSAGE);

fsizestr = String.valueOf(sizelist.getSelectedValue());

fsize = Integer.parseInt(fsizestr);

font1 = new Font(ffamily, fstyle, fsize);

maintext.setFont(font1);

} else if (ae.getSource() == exit) {

frame1.dispose();

} else if (ae.getSource() == savedoc) {

title.setText(topicstitle);

tle = title.getText();

try {

FileOutputStream filesave = new FileOutputStream(topicstitle + ".txt");

String s = maintext.getText();

for (int i = 0; i < s.length(); i++) {

filesave.write(s.charAt(i));

}

filesave.close();

} catch (Exception e) {

}

} else if (ae.getSource() == saveasdoc) {

if (title.getText().length() == 0) {

topicstitle = JOptionPane.showInputDialog(null, "Enter Your File Title?", "Your File Name",

JOptionPane.QUESTION\_MESSAGE);

title.setText(topicstitle);

tle = title.getText();

try {

FileOutputStream filesave = new FileOutputStream(tle + ".txt");

String s = maintext.getText();

for (int i = 0; i < s.length(); i++) {

frame1.setTitle(topicstitle + ".txt");

filesave.write(s.charAt(i));

}

filesave.close();

} catch (Exception e) {

}

}

} else if (ae.getSource() == opendoc) {

@SuppressWarnings("unused")

JFileChooser chooser = new JFileChooser();

}

}

public void keyTyped(KeyEvent ke) {

cl = maintext.getText().length();

linecount = maintext.getLineCount();

details.setText("Length: " + cl + " Line: " + linecount);

}

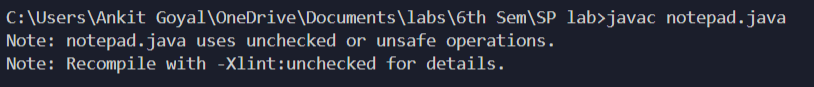
public static void main(String ar[]) {

new notepad();

}

}

**OUTPUT:**





**Practical-15**

**Aim: Design and implement one pass Assembler in any language.**

**THEORY:-**

**Assembler** is a program for converting instructions written in low-level assembly code into relocatable machine code and generating along information for the loader.

* One-pass assemblers are used when it is necessary or desirable to avoid a second pass over the source program the external storage for the intermediate file between two passes is slow or is inconvenient to use
* Main problem: forward references to both data and instructions
* One simple way to eliminate this problem: require that all areas be defined before they are referenced.
  + - It is possible, although inconvenient, to do so for data items.
    - Forward jump to instruction items cannot be easily eliminated.

**Algorithm for Pass 1 assembler:**

begin

if starting address is given  
LOCCTR = starting address;  
else  
LOCCTR = 0;  
while OPCODE != END do; or EOF  
begin  
read a line from the code  
if there is a label

if this label is in SYMTAB, then error  
else insert (label, LOCCTR) into SYMTAB  
search OPTAB for the op code  
if found

LOCCTR += N; N is the length of this instruction (4 for MIPS)  
else if this is an assembly directive

update LOCCTR as directed  
else error

write line to intermediate file  
end  
program size =  LOCCTR - starting address;  
end

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

char opcode[10],operand[10],label[10],a[10],ad[10],symbol[10],ch;

char code[10][10],code1[10][10]={"33","44","53","57"};

char mnemonic[10][10]={"START","LDA","STA","LDCH","STCH","END"};

char mnemonic1[10][10]={"LDA","STA","LDCH","STCH"};

int locctr,start,length,i=0,j=0,k,l=0;

int st,diff,address,add,len,actual\_len,finaddr,prevaddr;

FILE \*fp1,\*fp2,\*fp3,\*fp4,\*fp5,\*fp6,\*fp7;

clrscr();

fp1=fopen("INPUT.DAT","r");

fp2=fopen("SYMTAB.DAT","w");

fp3=fopen("INETERMED.DAT","w");

fscanf(fp1,"%s%s%s",label,opcode,operand);

if(strcmp(opcode,"START")==0)

{

start=atoi(operand);

locctr=start;

fprintf(fp3,"%s\t%s\t%s\n",label,opcode,operand);

fscanf(fp1,"%s%s%s",label,opcode,operand);

}

else

locctr=0;

while(strcmp(opcode,"END")!=0)

{

fprintf(fp3,"%d",locctr);

if(strcmp(label,"\*\*")!=0)

fprintf(fp2,"%s\t%d\n",label,locctr);

strcpy(code[i],mnemonic[j]);

while(strcmp(mnemonic[j],"END")!=0)

{

if(strcmp(opcode,mnemonic[j])==0)

{

locctr+=3;

break;

}

strcpy(code[i],mnemonic[j]);

j++;

}

if(strcmp(opcode,"WORD")==0)

locctr+=3;

else if(strcmp(opcode,"RESW")==0)

locctr+=(3\*(atoi(operand)));

else if(strcmp(opcode,"RESB")==0)

locctr+=(atoi(operand));

else if(strcmp(opcode,"BYTE")==0)

++locctr;

fprintf(fp3,"\t%s\t%s\t%s\n",label,opcode,operand);

fscanf(fp1,"%s%s%s",label,opcode,operand);

}

fprintf(fp3,"%d\t%s\t%s\t%s\n",locctr,label,opcode,operand);

length=locctr-start;

fcloseall();

printf("\n\nThe contents of Input file:\n\n");

fp1=fopen("INPUT.DAT","r");

ch=fgetc(fp1);

while(ch!=EOF)

{

printf("%c",ch);

ch=fgetc(fp1);

}

printf("\n\nLength of the input program is %d.",length);

printf("\n\nThe contents of Symbol Table:\n\n");

fp2=fopen("SYMTAB.DAT","r");

ch=fgetc(fp2);

while(ch!=EOF)

{

printf("%c",ch);

ch=fgetc(fp2);

}

fcloseall();

fp4=fopen("ASSMLIST.DAT","w");

fp5=fopen("SYMTAB.DAT","r");

fp6=fopen("INTERMED.DAT","r");

fp7=fopen("OBJCODE.DAT","w");

fscanf(fp6,"%s%s%s",label,opcode,operand);

while(strcmp(opcode,"END")!=0)

{

prevaddr=address;

fscanf(fp6,"%d%s%s%s",&address,label,opcode,operand);

}

finaddr=address;

fclose(fp6);

fp6=fopen("INTERMED.DAT","r");

fscanf(fp6,"%s%s%s",label,opcode,operand);

if(strcmp(opcode,"START")==0)

{

fprintf(fp4,"\t%s\t%s\t%s\n",label,opcode,operand);

fprintf(fp7,"H^%s^00%s^00%d\n",label,operand,finaddr);

fscanf(fp6,"%d%s%s%s",&address,label,opcode,operand);

st=address;

diff=prevaddr-st;

fprintf(fp7,"T^00%d^%d",address,diff);

}

while(strcmp(opcode,"END")!=0)

{

if(strcmp(opcode,"BYTE")==0)

{

fprintf(fp4,"%d\t%s\t%s\t%s\t",address,label,opcode,operand);

len=strlen(operand);

actual\_len=len-3;

fprintf(fp7,"^");

for(k=2;k<(actual\_len+2);k++)

{

itoa(operand[k],ad,16);

fprintf(fp4,"%s",ad);

fprintf(fp7,"%s",ad);

}

fprintf(fp4,"\n");

}

else if(strcmp(opcode,"WORD")==0)

{

len=strlen(operand);

itoa(atoi(operand),a,10);

fprintf(fp4,"%d\t%s\t%s\t%s\t00000%s\n",address,label,opcode,operand,a);

fprintf(fp7,"^00000%s",a);

}

else if((strcmp(opcode,"RESB")==0)||(strcmp(opcode,"RESW")==0))

fprintf(fp4,"%d\t%s\t%s\t%s\n",address,label,opcode,operand);

else

{

while(strcmp(opcode,mnemonic1[l])!=0)

l++;

if(strcmp(operand,"COPY")==0)

fprintf(fp4,"%d\t%s\t%s\t%s\t%s0000\n",address,label,opcode,operand,code1[l]);

else

{

rewind(fp5);

fscanf(fp5,"%s%d",symbol,&add);

while(strcmp(operand,symbol)!=0)

fscanf(fp5,"%s%d",symbol,&add);

fprintf(fp4,"%d\t%s\t%s\t%s\t%s%d\n",address,label,opcode,operand,code1[l],add);

fprintf(fp7,"^%s%d",code1[l],add);

}

}

fscanf(fp6,"%d%s%s%s",&address,label,opcode,operand);

}

fprintf(fp4,"%d\t%s\t%s\t%s\n",address,label,opcode,operand);

fprintf(fp7,"\nE^00%d",st);

printf("\nObject Program has been generated.");

fcloseall();

printf("\n\nObject Program:\n\n");

fp7=fopen("OBJCODE.DAT","r");

ch=fgetc(fp7);

while(ch!=EOF)

{

printf("%c",ch);

ch=fgetc(fp7);

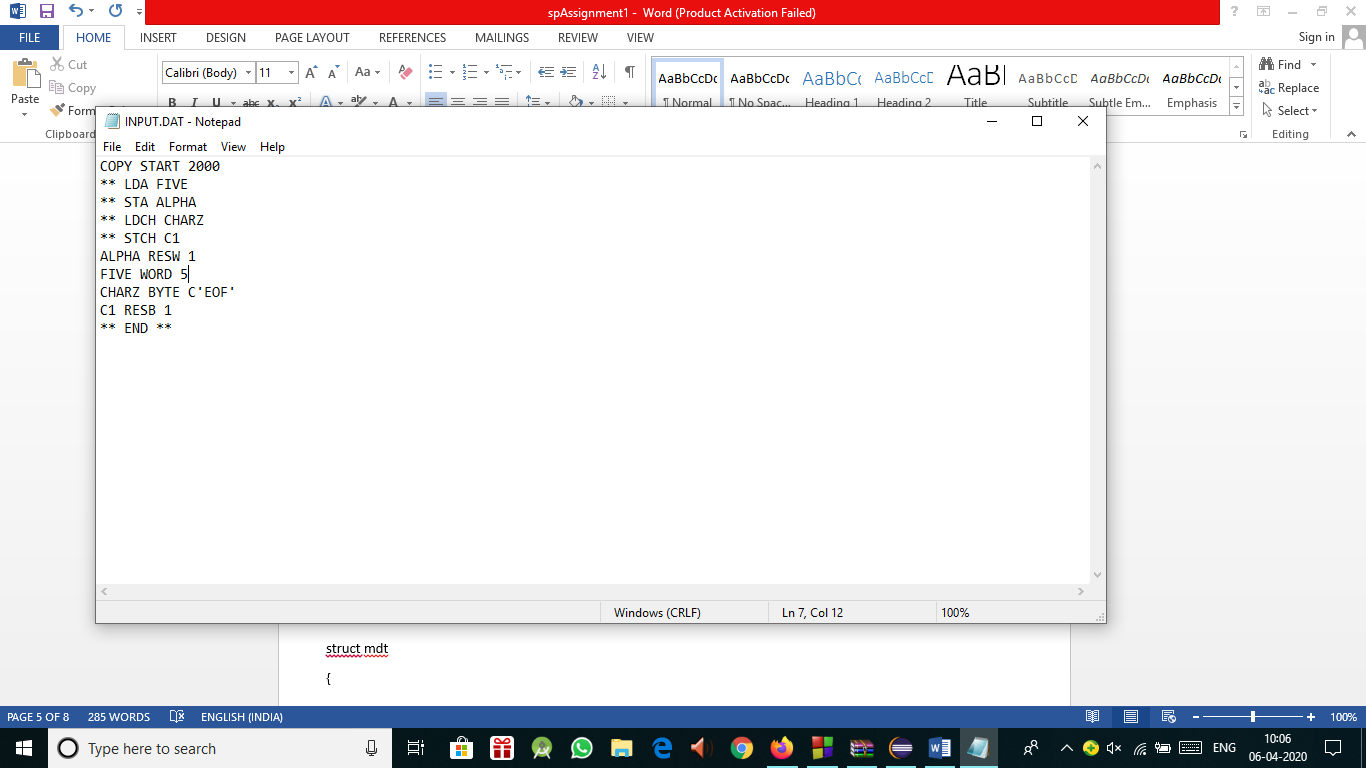
}

fcloseall();

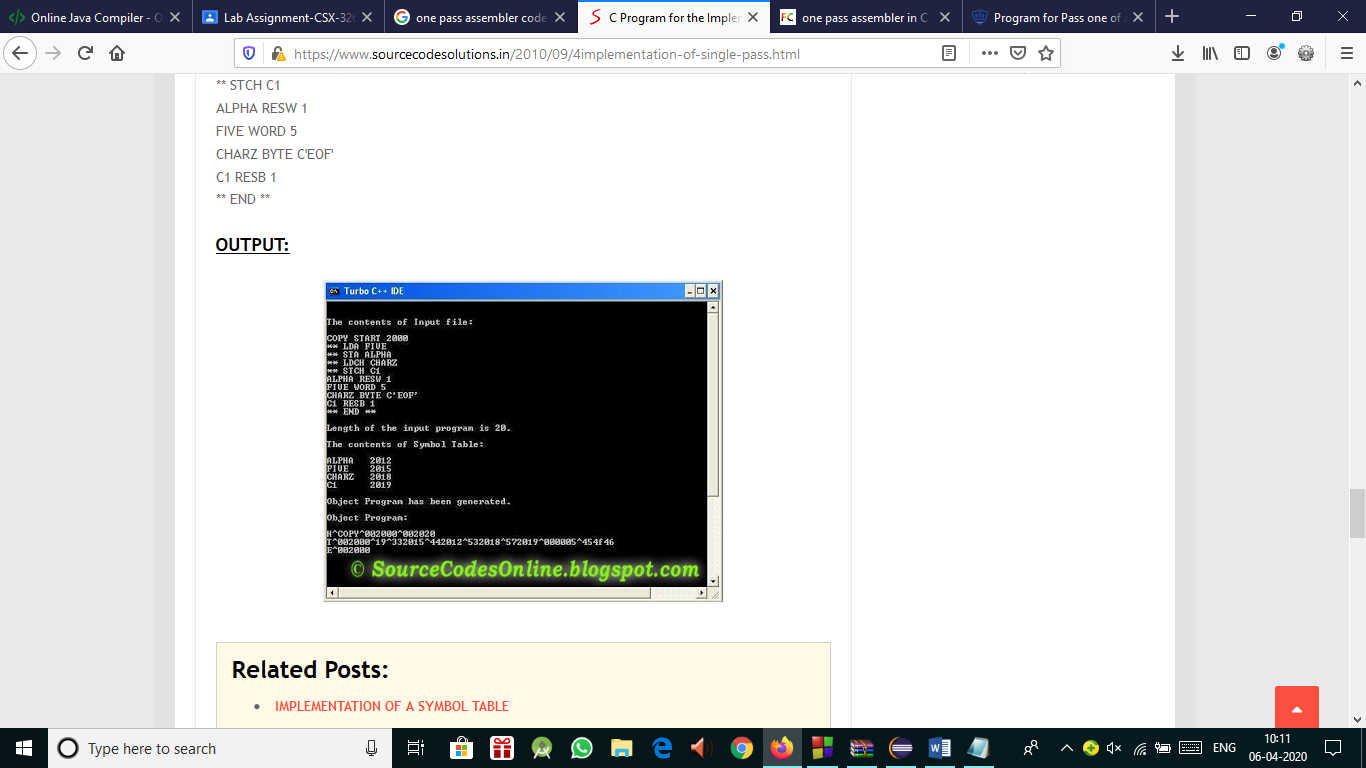
getch();

}

**INPUT:**



**OUTPUT:**



**Practical-16**

**Aim: Design and implement two pass assembler in any language.**

**THEORY :**

**Two-pass assembler:**Assemblers typically make two or more passes through a source program in order to resolve forward references in a program. A forward reference is defined as a type of instruction in the code segment that is referencing the label of an instruction, but the assembler has not yet encountered the definition of that instruction.

**Pass 1:**Assembler reads the entire source program and constructs a symbol table of names and labels used in the program, that is, name of data fields and programs labels and their relative location (offset) within the segment.

Pass 1 determines the amount of code to be generated for each instruction.

**Pass 2:**The assembler uses the symbol table that it constructed in Pass 1. Now it knows the length and relative of each data field and instruction, it can complete the object code for each instruction. It produces .OBJ (Object file), .LST (list file) and cross reference (.CRF) files.

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

char a[10],ad[10],label[10],opcode[10],operand[10],symbol[10],ch; int st,diff,i,address,add,len,actual\_len,finaddr,prevaddr,j=0;

char mnemonic[15][15]={"LDA","STA","LDCH","STCH"};

char code[15][15]={"33","44","53","57"};

FILE \*fp1,\*fp2,\*fp3,\*fp4;

clrscr();

fp1=fopen("ASSMLIST.DAT","w");

fp2=fopen("SYMTAB.DAT","r");

fp3=fopen("INTERMED.DAT","r");

fp4=fopen("OBJCODE.DAT","w");

fscanf(fp3,"%s%s%s",label,opcode,operand);

while(strcmp(opcode,"END")!=0)

{

prevaddr=address;

fscanf(fp3,"%d%s%s%s",&address,label,opcode,operand);

}

finaddr=address;

fclose(fp3);

fp3=fopen("INTERMED.DAT","r");

fscanf(fp3,"%s%s%s",label,opcode,operand);

if(strcmp(opcode,"START")==0)

{

fprintf(fp1,"\t%s\t%s\t%s\n",label,opcode,operand);

fprintf(fp4,"H^%s^00%s^00%d\n",label,operand,finaddr);

fscanf(fp3,"%d%s%s%s",&address,label,opcode,operand);

st=address;

diff=prevaddr-st;

fprintf(fp4,"T^00%d^%d",address,diff);

}

while(strcmp(opcode,"END")!=0)

{

if(strcmp(opcode,"BYTE")==0)

{

fprintf(fp1,"%d\t%s\t%s\t%s\t",address,label,opcode,operand);

len=strlen(operand);

actual\_len=len-3;

fprintf(fp4,"^");

for(i=2;i<(actual\_len+2);i++)

{

itoa(operand[i],ad,16);

fprintf(fp1,"%s",ad);

fprintf(fp4,"%s",ad);

}

fprintf(fp1,"\n");

}

else if(strcmp(opcode,"WORD")==0)

{

len=strlen(operand);

itoa(atoi(operand),a,10);

fprintf(fp1,"%d\t%s\t%s\t%s\t00000%s\n",address,label,opcode,operand,a);

fprintf(fp4,"^00000%s",a);

}

else if((strcmp(opcode,"RESB")==0)||(strcmp(opcode,"RESW")==0))

fprintf(fp1,"%d\t%s\t%s\t%s\n",address,label,opcode,operand);

else

{

while(strcmp(opcode,mnemonic[j])!=0)

j++;

if(strcmp(operand,"COPY")==0)

fprintf(fp1,"%d\t%s\t%s\t%s\t%s0000\n",address,label,opcode,operand,code[j]);

else

{

rewind(fp2);

fscanf(fp2,"%s%d",symbol,&add);

while(strcmp(operand,symbol)!=0)

fscanf(fp2,"%s%d",symbol,&add);

fprintf(fp1,"%d\t%s\t%s\t%s\t%s%d\n",address,label,opcode,operand,code[j],add);

fprintf(fp4,"^%s%d",code[j],add);

}

}

fscanf(fp3,"%d%s%s%s",&address,label,opcode,operand);

}

fprintf(fp1,"%d\t%s\t%s\t%s\n",address,label,opcode,operand);

fprintf(fp4,"\nE^00%d",st);

printf("\n Intermediate file is converted into object code");

fcloseall();

printf("\n\nThe contents of Intermediate file:\n\n\t");

fp3=fopen("INTERMED.DAT","r");

ch=fgetc(fp3);

while(ch!=EOF)

{

printf("%c",ch);

ch=fgetc(fp3);

}

printf("\n\nThe contents of Symbol Table :\n\n");

fp2=fopen("SYMTAB.DAT","r");

ch=fgetc(fp2);

while(ch!=EOF)

{

printf("%c",ch);

ch=fgetc(fp2);

}

printf("\n\nThe contents of Output file :\n\n");

fp1=fopen("ASSMLIST.DAT","r");

ch=fgetc(fp1);

while(ch!=EOF)

{

printf("%c",ch);

ch=fgetc(fp1);

}

printf("\n\nThe contents of Object code file :\n\n");

fp4=fopen("OBJCODE.DAT","r");

ch=fgetc(fp4);

while(ch!=EOF)

{

printf("%c",ch);

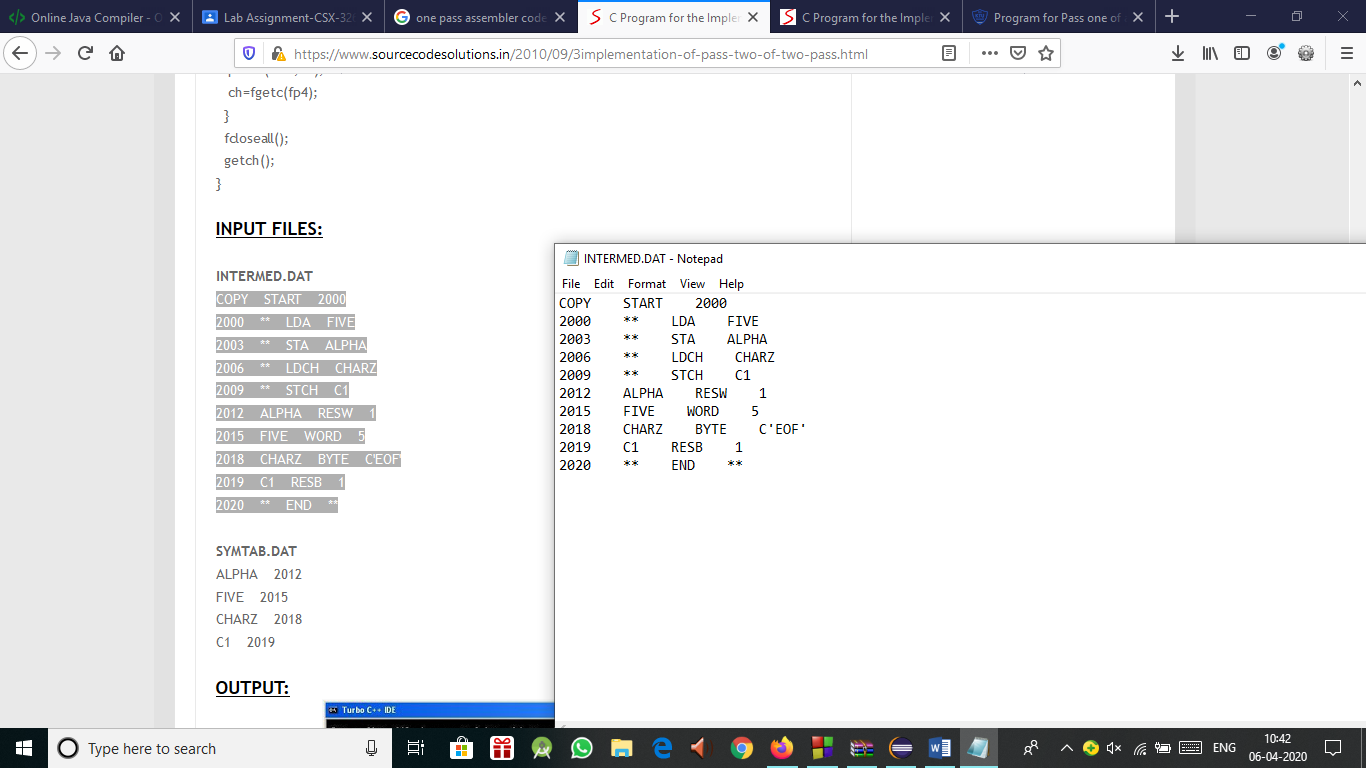
ch=fgetc(fp4);

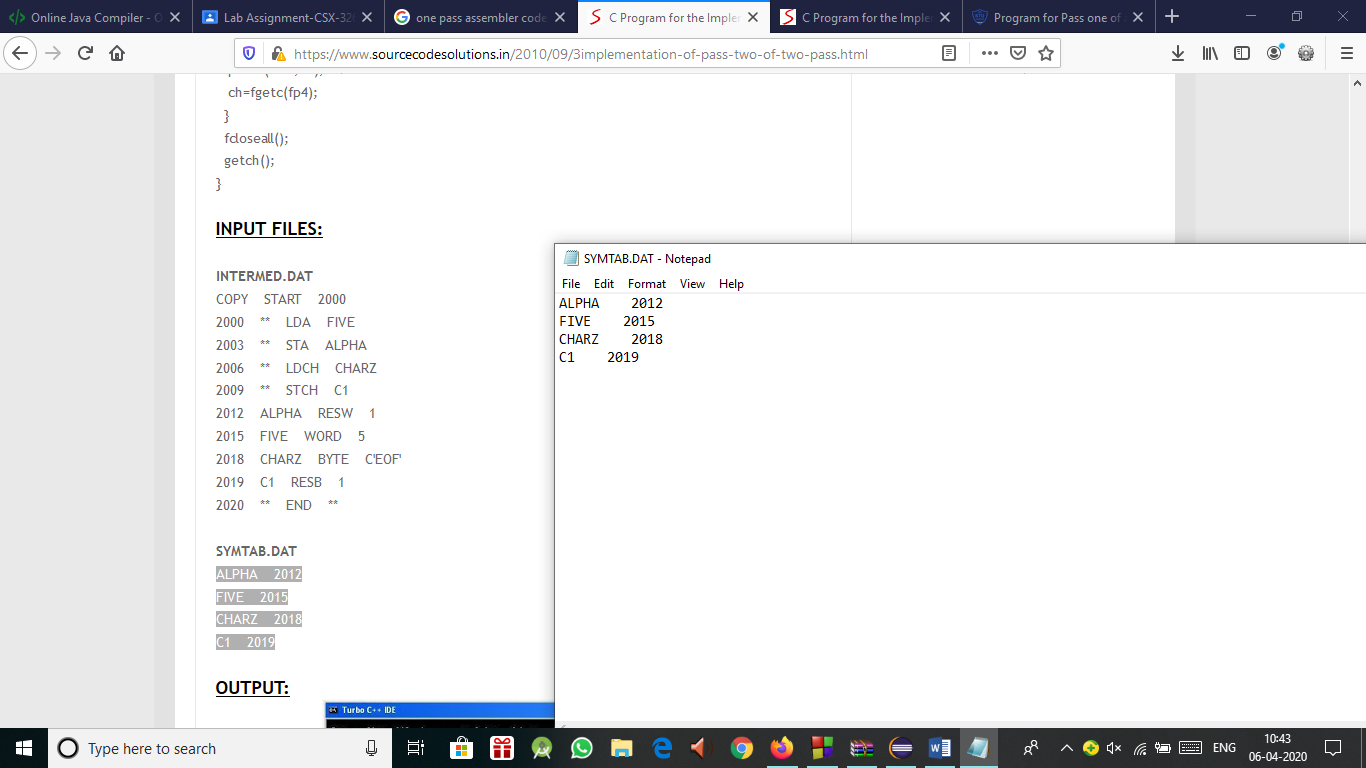
}

fcloseall();

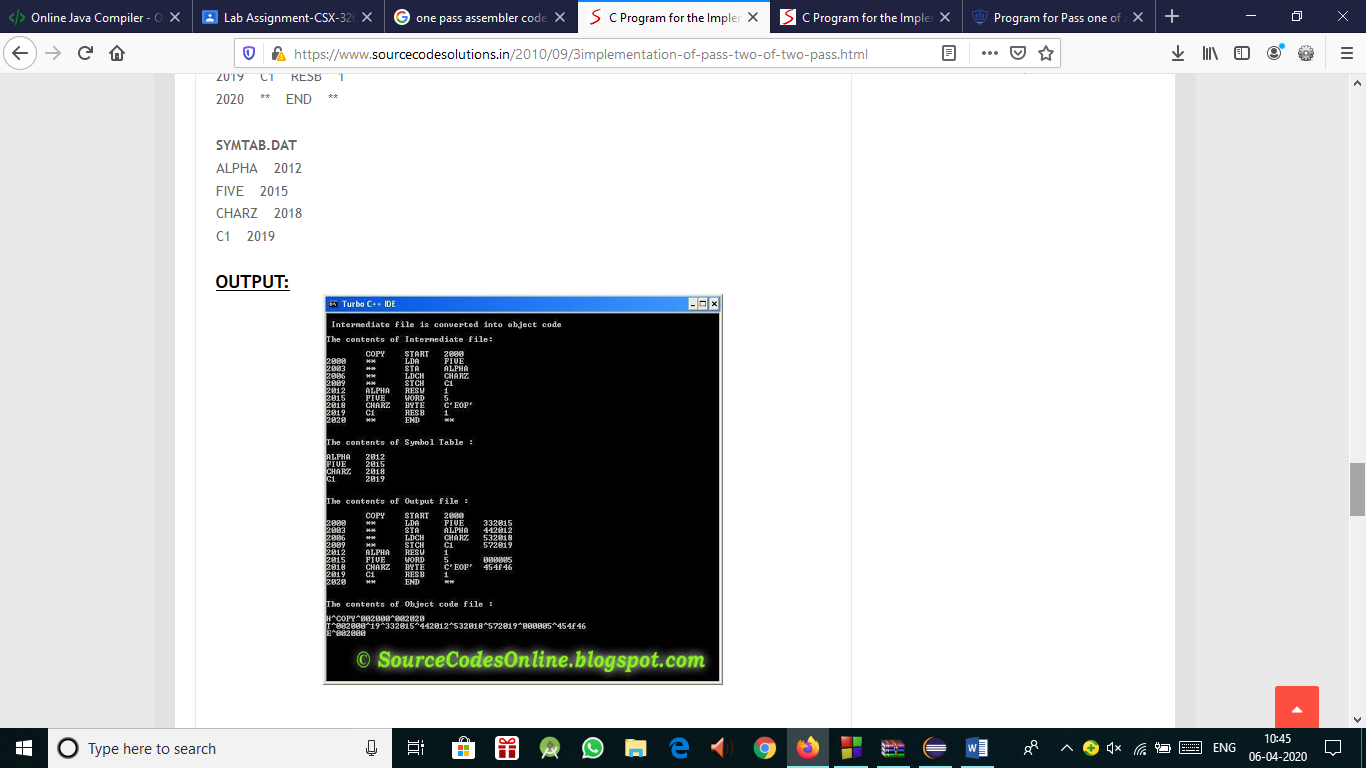
getch();}

**INPUT :**





**OUTPUT :**



**Practical-17**

**Aim: Design and implement of two pass macro processor.**

**THEORY :**

A Macro instruction is the notational convenience for the programmer. For every occurrence of macro the whole macro body or macro block of statements gets expanded in the main source code. Thus Macro instructions makes writing code more convenient.

**Salient features of Macro Processor:**

* **Macro** represents a group of commonly used statements in the source programming language.
* Macro Processors replace each macro instruction with the corresponding group of source language statements. This is known as expansion of macros.
* Using Macro instructions programmers can leave the mechanical details to be handled by the macro processor.
* Macro Processor designs are not directly related to the computer architecture on which it runs.
* Macro Processor involves definition, invocation and expansion.
* **Two-pass macro processor** : All macro definitions are processed during the first pass.
* All macro invocation statements are expanded during the second pass.
* Two-pass macro processor would not allow the body of one macro instruction to contain definitions of other macros.

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

struct mdt

{

char lab[10];

char opc[10];

char oper[10];

}d[10];

void main()

{

char label[10],opcode[10],operand[10],newlabel[10],newoperand[10];

char macroname[10];

int i,lines;

FILE \*f1,\*f2,\*f3;

clrscr();

f1 = fopen("MACIN.txt","r");

f2 = fopen("MACOUT.txt","w");

f3 = fopen("MDT.txt","w");

fscanf(f1,"%s %s %s",label,opcode,operand);

while(strcmp(opcode,"END")!=0)

{

if(strcmp(opcode,"MACRO")==0)

{

strcpy(macroname,label);

fscanf(f1,"%s%s%s",label,opcode,operand);

lines = 0;

while(strcmp(opcode,"MEND")!=0)

{

fprintf(f3,"%s\t%s\t%s\n",label,opcode,operand);

strcpy(d[lines].lab,label);

strcpy(d[lines].opc,opcode);

strcpy(d[lines].oper,operand);

fscanf(f1,"%s %s %s",label,opcode,operand);

lines++;

}

}

else if(strcmp(opcode,macroname)==0)

{

printf("lines=%d\n",lines);

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

{

fprintf(f2,"%s\t%s\t%s\n",d[i].lab,d[i].opc,d[i].oper);

printf("DLAB=%s\nDOPC=%\nDOPER=%s\n",d[i].lab,d[i].opc,d[i].oper);

}

}

else

fprintf(f2,"%s\t%s\t%s\n",label,opcode,operand);

fscanf(f1,"%s%s%s",label,opcode,operand);

}

fprintf(f2,"%s\t%s\t%s\n",label,opcode,operand);

fclose(f1);

fclose(f2);

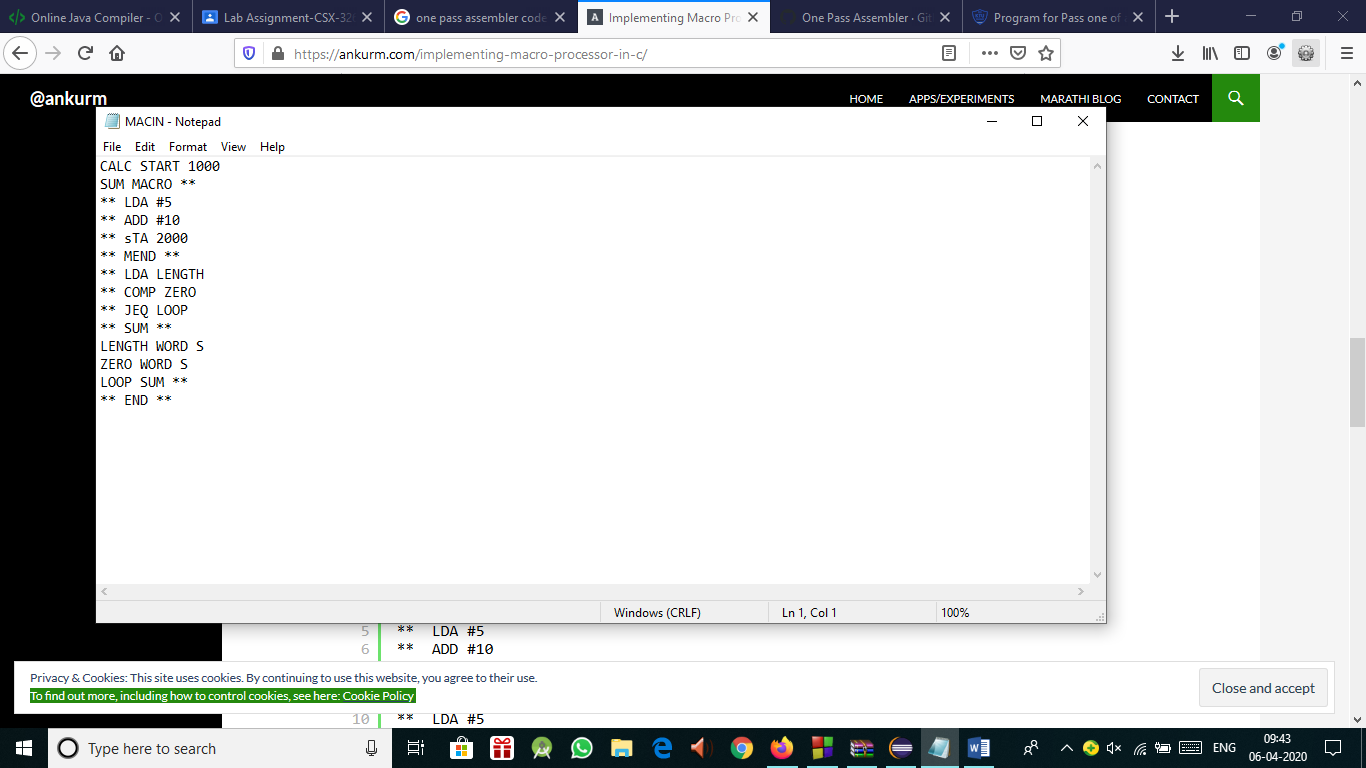
fclose(f3);

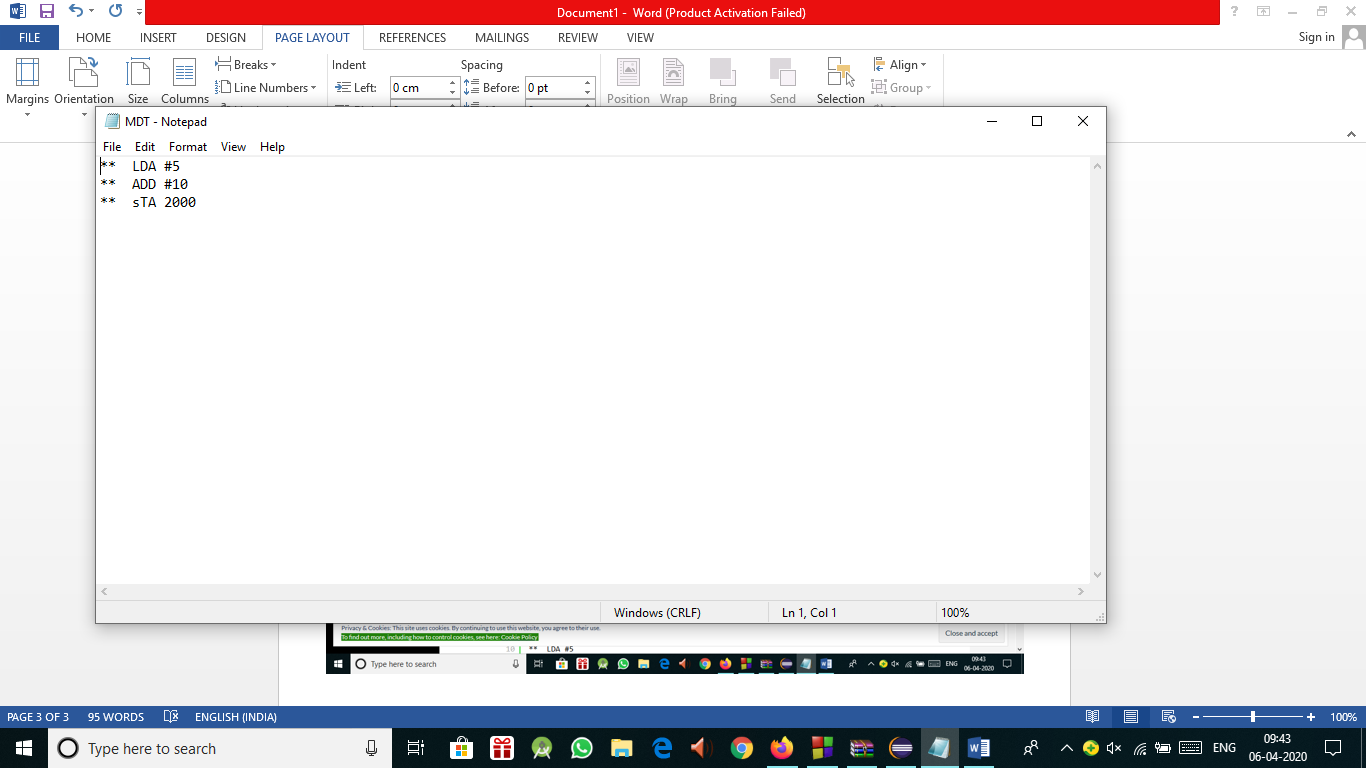
printf("FINISHED");

getch();

}

**INPUT:**





**OUTPUT:**

