**Compiler Design**

* **Assignment – 9: Generate 3-tuple intermediate code for given infix expression**

**CODE:**

**#include<bits/stdc++.h>**

**#include<string>**

**using namespace std;**

**char stac[20],val1[20],sym[20];**

**int val[20];**

**int top1=-1,top2=-1,top3=-1;**

**string input;**

**void print\_stac(){**

**for(int i=0;i<=top1;i++){**

**cout<<stac[i];**

**}**

**}**

**void print\_val(){**

**for(int i=0;i<=top2;i++){**

**cout<<val[i]<<" ";**

**}**

**}**

**void print\_val1(){**

**for(int i=0;i<=top2;i++){**

**cout<<val1[i];**

**}**

**}**

**void sdt(){**

**stac[0] = '$';top1=0;**

**input[input.length()]='$';**

**cout<<"Stack\tValue\n---------------\n";**

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

**if(input[i]>='0' && input[i]<='9'){**

**stac[top1+1] = 'd';**

**top1+=1;**

**val[top2+1] = int(input[i])-48;**

**top2+=1;**

**print\_stac();**

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

**print\_val();**

**cout<<endl;**

**}**

**else{**

**stac[top1+1] = input[i];**

**top1+=1;**

**print\_stac();**

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

**print\_val();**

**cout<<endl;**

**}**

**if(stac[top1]=='d' && stac[top1-1]=='I'){**

**stac[top1-1] = 'I';**

**top1-=1;**

**val[top2-1] = 10\*val[top2-1] + val[top2];**

**top2-=1;**

**print\_stac();**

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

**print\_val();**

**cout<<endl;**

**}**

**if(stac[top1]=='d'){**

**stac[top1]='I';**

**print\_stac();**

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

**print\_val();**

**cout<<endl;**

**}**

**if(stac[top1]=='I' && (input[i+1]<'0' || input[i+1]>'9')){**

**stac[top1] = 'E';**

**print\_stac();**

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

**print\_val();**

**cout<<endl;**

**}**

**if(stac[top1]==')' && stac[top1-1]=='E' && stac[top1-2]=='('){**

**stac[top1-2]='E';**

**top1-=2;**

**print\_stac();**

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

**print\_val();**

**cout<<endl;**

**}**

**if(stac[top1]=='E' && stac[top1-1]=='+' && stac[top1-2]=='E'){**

**stac[top1-2]='E';**

**top1-=2;**

**val[top2-1] = val[top2]+val[top2-1];**

**top2-=1;**

**print\_stac();**

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

**print\_val();**

**cout<<endl;**

**}**

**if(stac[top1]=='E' && stac[top1-1]=='\*' && stac[top1-2]=='E'){**

**stac[top1-2]='E';**

**top1-=2;**

**val[top2-1] = val[top2]\*val[top2-1];**

**top2-=1;**

**print\_stac();**

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

**print\_val();**

**cout<<endl;**

**}**

**if(stac[top1]=='$' && stac[top1-1]=='E'){**

**stac[top1-1]='S';**

**top1-=1;**

**print\_stac();**

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

**print\_val();**

**cout<<endl;**

**}**

**}**

**cout<<val[top2]<<endl;**

**}**

**void convert(){**

**stac[0] = '$';top1=0;top2=-1;**

**input[input.length()]='$';**

**cout<<"Stack\tPost-fix\n---------------\n";**

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

**if(input[i]>='0' && input[i]<='9'){**

**stac[top1+1] = 'd';**

**top1+=1;**

**val1[top2+1] = input[i];**

**top2+=1;**

**print\_stac();**

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

**print\_val1();**

**cout<<endl;**

**}**

**else{**

**stac[top1+1] = input[i];**

**if(input[i]=='\*' || input[i]=='+'){**

**sym[top3+1]=input[i];**

**top3+=1;**

**}**

**top1+=1;**

**print\_stac();**

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

**print\_val1();**

**cout<<endl;**

**}**

**if(stac[top1]=='d' && stac[top1-1]=='I'){**

**stac[top1-1] = 'I';**

**top1-=1;**

**print\_stac();**

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

**print\_val1();**

**cout<<endl;**

**}**

**if(stac[top1]=='d'){**

**stac[top1]='I';**

**print\_stac();**

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

**print\_val1();**

**cout<<endl;**

**}**

**if(stac[top1]=='I' && (input[i+1]<'0' || input[i+1]>'9')){**

**stac[top1] = 'E';**

**print\_stac();**

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

**print\_val1();**

**cout<<endl;**

**}**

**if(stac[top1]==')' && stac[top1-1]=='E' && stac[top1-2]=='('){**

**stac[top1-2]='E';**

**top1-=2;**

**val1[top2+1]=sym[top3];**

**top3-=1;top2+=1;**

**print\_stac();**

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

**print\_val1();**

**cout<<endl;**

**}**

**if(stac[top1]=='E' && stac[top1-1]=='+' && stac[top1-2]=='E'){**

**stac[top1-2]='E';**

**top1-=2;**

**print\_stac();**

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

**print\_val1();**

**cout<<endl;**

**}**

**if(stac[top1]=='E' && stac[top1-1]=='\*' && stac[top1-2]=='E'){**

**stac[top1-2]='E';**

**top1-=2;**

**print\_stac();**

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

**print\_val1();**

**cout<<endl;**

**}**

**if(stac[top1]=='$' && stac[top1-1]=='E'){**

**stac[top1-1]='S';**

**top1-=1;**

**print\_stac();**

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

**print\_val1();**

**cout<<endl;**

**}**

**}**

**print\_stac();**

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

**print\_val1();**

**while(top3!=-1){**

**cout<<sym[top3];**

**top3-=1;**

**}**

**cout<<endl;**

**}**

**void threeAddressCode(){**

**stac[0] = '$';top1=0;top2=-1;**

**int x=1;**

**vector<vector<string> > v;**

**input[input.length()]='$';**

**cout<<"Stack\tPlace\tGenerated Code\n---------------------------------\n";**

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

**if(input[i]>='0' && input[i]<='9'){**

**stac[top1+1] = 'd';**

**top1+=1;**

**val[top2+1] = int(input[i])-48;**

**top2+=1;**

**print\_stac();**

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

**print\_val();**

**cout<<endl;**

**}**

**else{**

**stac[top1+1] = input[i];**

**top1+=1;**

**print\_stac();**

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

**print\_val();**

**cout<<endl;**

**}**

**if(stac[top1]=='d' && stac[top1-1]=='I'){**

**stac[top1-1] = 'I';**

**top1-=1;**

**val[top2-1] = 10\*val[top2-1] + val[top2];**

**top2-=1;**

**print\_stac();**

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

**print\_val();**

**cout<<endl;**

**}**

**if(stac[top1]=='d'){**

**stac[top1]='I';**

**print\_stac();**

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

**print\_val();**

**cout<<endl;**

**}**

**if(stac[top1]=='I' && (input[i+1]<'0' || input[i+1]>'9')){**

**stac[top1] = 'E';**

**print\_stac();**

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

**print\_val();**

**cout<<endl;**

**}**

**if(stac[top1]==')' && stac[top1-1]=='E' && stac[top1-2]=='('){**

**stac[top1-2]='E';**

**top1-=2;**

**print\_stac();**

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

**print\_val();**

**cout<<endl;**

**}**

**if(stac[top1]=='E' && stac[top1-1]=='+' && stac[top1-2]=='E'){**

**print\_stac();**

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

**print\_val();**

**if(x>1)**

**cout<<"\tT"<<x<<" := "<<val[top2-1]<<" + T"<<x-1;**

**else**

**cout<<"\tT"<<x<<" := "<<val[top2-1]<<" + "<<val[top2];**

**x++;**

**cout<<endl;**

**stac[top1-2]='E';**

**top1-=2;**

**val[top2-1] = val[top2]+val[top2-1];**

**top2-=1;**

**print\_stac();**

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

**print\_val();**

**cout<<endl;**

**}**

**if(stac[top1]=='E' && stac[top1-1]=='\*' && stac[top1-2]=='E'){**

**print\_stac();**

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

**print\_val();**

**if(x>1)**

**cout<<"\tT"<<x<<" := "<<val[top2-1]<<" \* T"<<x-1;**

**else**

**cout<<"\tT"<<x<<" := "<<val[top2-1]<<" \* "<<val[top2];**

**x++;**

**cout<<endl;**

**stac[top1-2]='E';**

**top1-=2;**

**val[top2-1] = val[top2]\*val[top2-1];**

**top2-=1;**

**print\_stac();**

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

**print\_val();**

**cout<<endl;**

**}**

**if(stac[top1]=='$' && stac[top1-1]=='E'){**

**stac[top1-1]='S';**

**top1-=1;**

**print\_stac();**

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

**print\_val();**

**cout<<endl;**

**}**

**}**

**}**

**int main(){**

**cout<<"Enter the input : ";**

**cin>>input;**

**cout<<"Syntax Directed Translation\n================================\n";**

**sdt();**

**cout<<"Infix to postfix\n================================\n";**

**convert();**

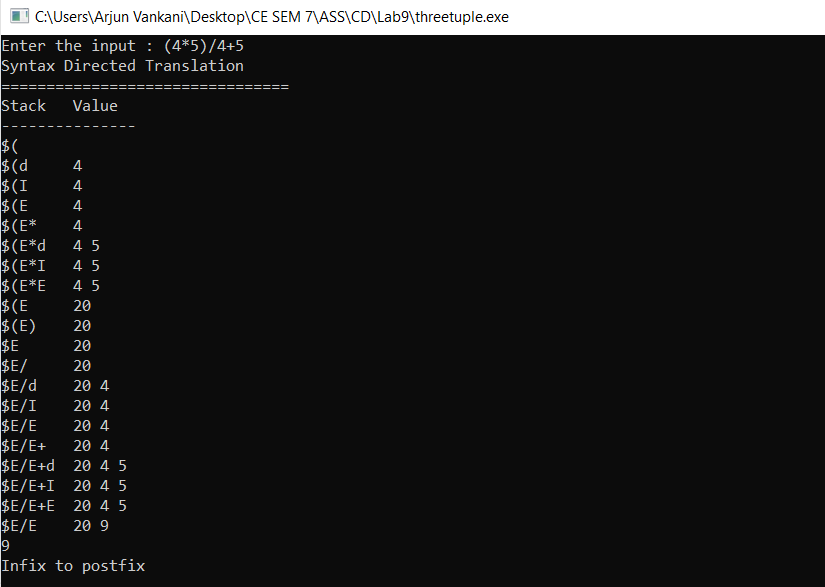
**cout<<"Three Address Code\n================================\n";**

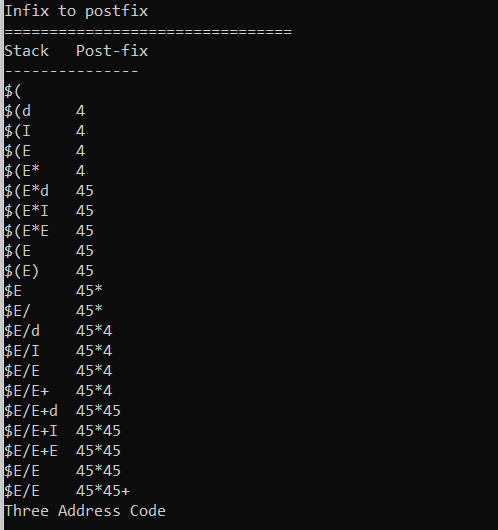
**threeAddressCode();**

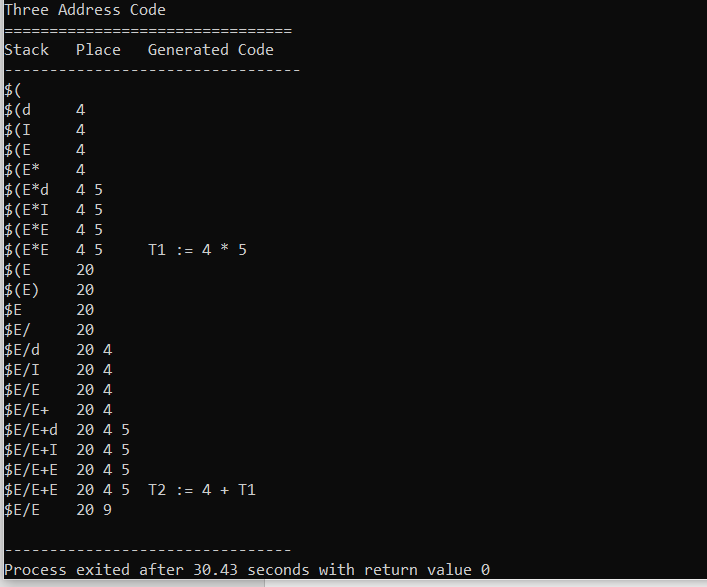
**return 0;**

**}**

**Output:**

****

****

****