Sethi Ullman Algorithm Implementation in c++
Ravi Jain (411764)

Method:
1. Input taken
2. Convert the input to binary expression tree (AST)
3. labeling done (Sethi Ullman algorithm phase 1)
4. Code Generation done on labelled expression tree.

Rules For Input:
1. Opening and closing parenthesis to be used for each sub block. (check the output images for further reference)

Code:

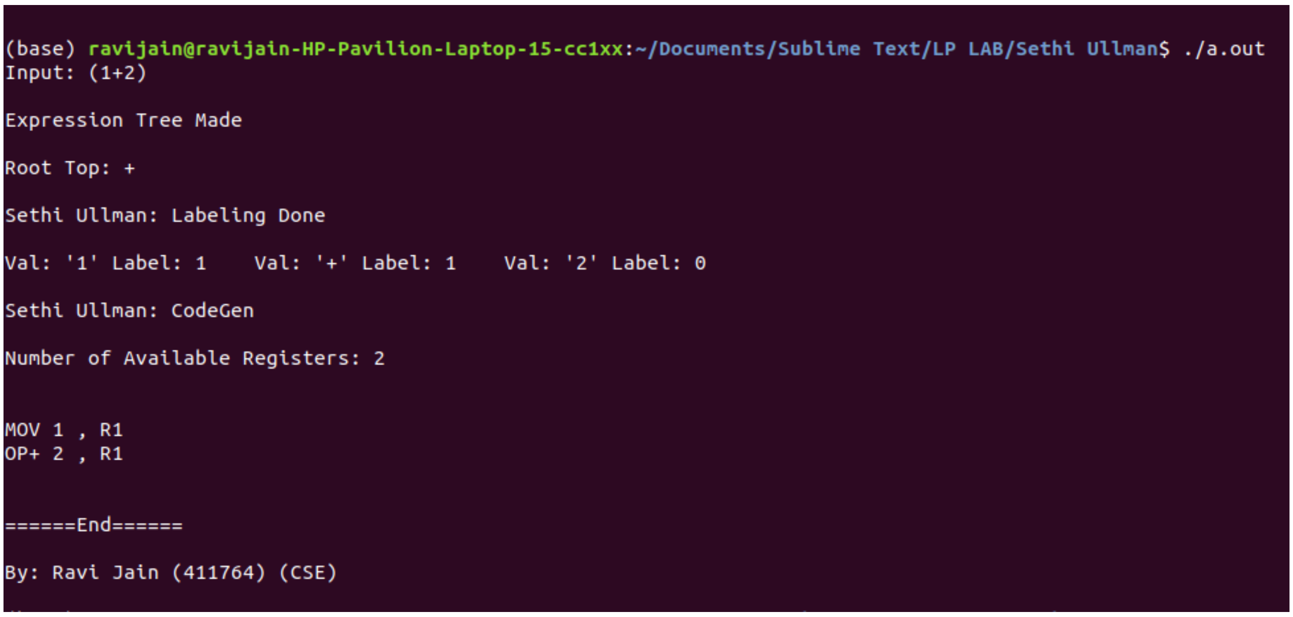
```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;

struct Node {
	char val=' ';
	int label=0;
	Node *leftNode=NULL;
	Node *rightNode=NULL;
};

string input() {
	string input;
	cout << "Input: ";
	cin >> input;
	bool check=true;
	for(int i=0; i<input.length(); i++) {
		if(!((int(input[i])>=48 && int(input[i])<=57) || input[i]=='+' || input[i]=='-' ||
input[i]=='(' || input[i]==')' || input[i]=='/' || input[i]=='*')) {
			check=false;
			break;
		}
	}
	if(input.length()==0 || !check) {
		cout << "\nInvalid Input\n";
		exit(0);
	}
	return input;
}

stack<Node*> local;
stack<Node*> global;

void reduce(bool Global, bool Local) {
	if(Global) {
			if(global.size()==3) {
				Node *right=(global.top());
				global.pop();
				Node *root=(global.top());
				global.pop();
				Node *left=(global.top());
				global.pop();
				(root)->leftNode=(left);
				(root)->rightNode=(right);
				global.push(root);
			}
		}
		if(Local) {
			if(local.size()==3) {
				Node *right=(local.top());
				local.pop();
				Node *root=(local.top());
				local.pop();
				Node *left=(local.top());
				local.pop();
				(root)->leftNode=(left);
				(root)->rightNode=(right);
				global.push(root);
			}
			else if(local.size()==1) {
				global.push(local.top());
				local.pop();
			}
		}
}



void printTree(Node *root) {
		if(root==NULL) {
			return;
		}
		if(root->leftNode) {
			printTree(root->leftNode);
		}
		cout << "Val: '" << root->val << "' Label: " << root->label << "     ";
		if(root->rightNode) {
			printTree(root->rightNode);
		}
}

void label(Node *root, bool left) {
		if(root==NULL) {
			return;
		}
		if(root->leftNode) {
			label(root->leftNode, true);
```

```
	}
	if(root->rightNode) {
		label(root->rightNode, false);
	}
	if(!root->leftNode && !root->rightNode) {
		if(left) {
			root->label=1;
		}
		else {
			root->label=0;
		}
	}
	else {
		int left=root->leftNode->label;
		int right=root->rightNode->label;
		if(left== right) {
			root->label=left+1;
		}
		else {
			root->label=max(left,right);
		}
	}
}

int NoReg=2, NoTemp=100;
int noRegUsed=0;
int NoTempUsed=0;
void codeGen(Node *root, bool left) {
	if(root==NULL) {
		cout << "\nCheck 1 \n";
		return;
	}
	//case 1
	if(!root->leftNode && !root->rightNode) {
		if(left) {
			noRegUsed++;
			cout << "MOV " << root->val << " , R" << noRegUsed << "\n";
		}
	}
	else if(root && root->rightNode->label==0) {
		codeGen(root->leftNode, true);
		cout << "OP" << root->val << "  " << root->rightNode->val  << " , R" <<
noRegUsed << "\n";
	}
	//case 3
	else if((1<=root->leftNode->label) && (root->leftNode->label < root->rightNode-
>label)
			&& (root->label <= (NoReg-noRegUsed))) {
		codeGen(root->rightNode, false);
		codeGen(root->leftNode, true);
		noRegUsed++;
		cout << "OP" << root->val << " R" << noRegUsed-1 << " , R" << noRegUsed
<< "\n";
		noRegUsed--;
	}
	else if((1<=root->rightNode->label) && (root->rightNode->label<=root->leftNode-
>label)
		&& (root->label <= (NoReg-noRegUsed))) {
		codeGen(root->leftNode, true);
		//noRegUsed++;
		codeGen(root->rightNode, true);
		cout << "OP" << root->val << " R" << noRegUsed << " , R" << noRegUsed -1
<< "\n";
		noRegUsed--;
	}
	else if((root->leftNode->label > (NoReg-noRegUsed)) && (root->rightNode->label >
(NoReg-noRegUsed))) {
		NoTempUsed++;
		codeGen(root->rightNode, false);
		cout << "MOV R" << noRegUsed << " ,T" << NoTempUsed << "\n";
		codeGen(root->leftNode, true);
		cout << "MOV T" << NoTempUsed << " , R" << noRegUsed << "\n";
		NoTempUsed--;
	}
	else {
		cout << "\nError\n";
	}
}

int main() {
	Node root;
	string in=input();
	stack<char> paren;
	int pos=0;
	do {
		if(global.size()==3) {
			reduce(true,false);
		}
		if(in[pos]=='(') {
			paren.push(in[pos++]);
		}
		else if(in[pos]!=')') {
			Node *node=new Node;
			node->val=in[pos++];
			local.push(node);
		}
		else if(in[pos]==')') {
			paren.pop();
			pos++;
			reduce(false,true);
			Node *node=new Node;
			if(in[pos]=='\0') {
				break;
```

```
                        }
                        if(in[pos]==')') {
                                continue;
                        }
                        node->val=in[pos++];
                        global.push(node);
                }
        } while(!paren.empty() && pos<in.size());
        root=*global.top();
        cout << "\nExpression Tree Made\n";
        cout << "\nRoot Top: " << root.val << "\n";
        label(&root, true);
        cout << "\nSethi Ullman: Labeling Done\n\n";
        printTree(&root);
        cout << "\n\nSethi Ullman: CodeGen\n\n";
        cout << "Number of Available Registers: ";
        cin >> NoReg;
        cout << "\n\n";
        if(NoReg<2) {
                cout << "Sorry Number of Registers can't be less than 2; Default Value of 2
used\n\n";
                NoReg=2;
        }
        codeGen(&root,true);
        cout << "\n\n======End======\n\nBy: Ravi Jain (411764) (CSE)\n\n";
        return 0;
}
```

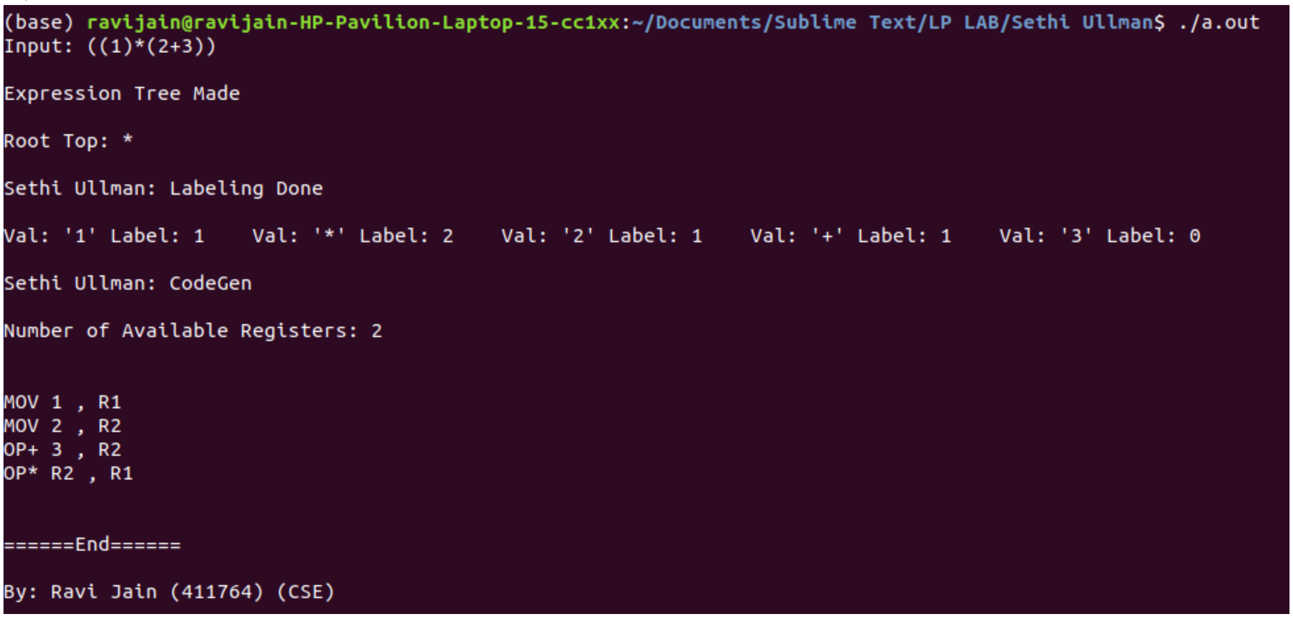
Output:

1.



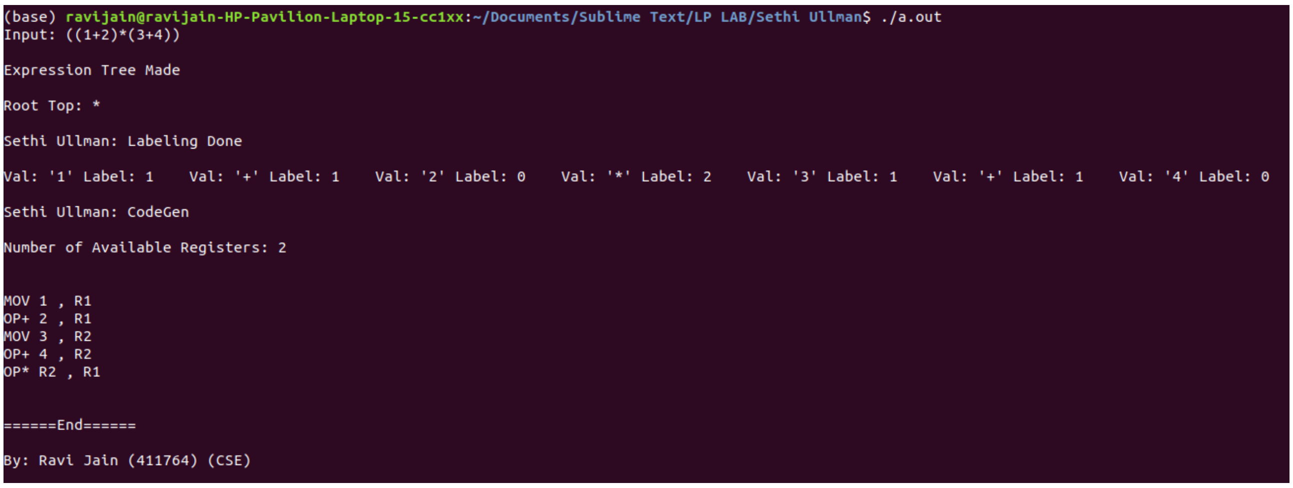
```
(base) ravijain@ravijain-HP-Pavilion-Laptop-15-cc1xx:~/Documents/Sublime Text/LP LAB/Sethi Ullman$ ./a.out
Input: (1+2)

Expression Tree Made

Root Top: +

Sethi Ullman: Labeling Done

Val: '1' Label: 1    Val: '+' Label: 1    Val: '2' Label: 0

Sethi Ullman: CodeGen

Number of Available Registers: 2


MOV 1 , R1
OP+ 2 , R1


======End======

By: Ravi Jain (411764) (CSE)
```

2.



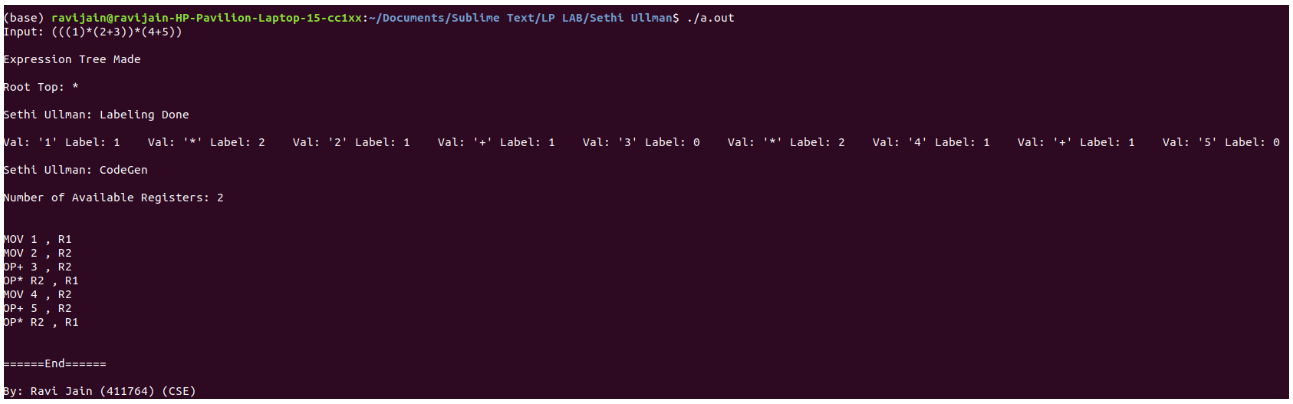
```
(base) ravijain@ravijain-HP-Pavilion-Laptop-15-cc1xx:~/Documents/Sublime Text/LP LAB/Sethi Ullman$ ./a.out
Input: ((1)*(2+3))

Expression Tree Made

Root Top: *

Sethi Ullman: Labeling Done

Val: '1' Label: 1    Val: '*' Label: 2    Val: '2' Label: 1    Val: '+' Label: 1    Val: '3' Label: 0

Sethi Ullman: CodeGen

Number of Available Registers: 2


MOV 1 , R1
MOV 2 , R2
OP+ 3 , R2
OP* R2 , R1


======End======

By: Ravi Jain (411764) (CSE)
```

3.



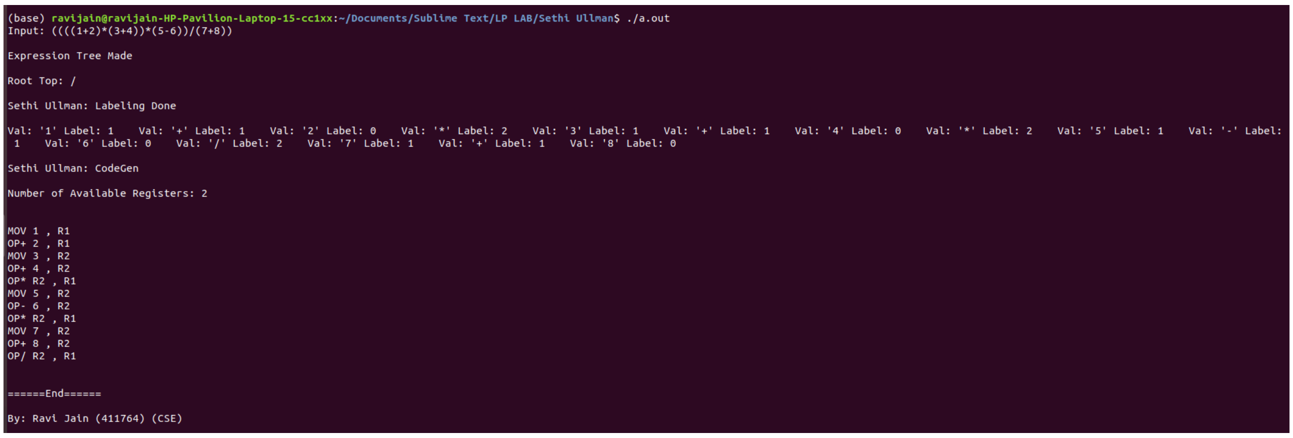
```
(base) ravijain@ravijain-HP-Pavilion-Laptop-15-cc1xx:~/Documents/Sublime Text/LP LAB/Sethi Ullman$ ./a.out
Input: ((1+2)*(3+4))

Expression Tree Made

Root Top: *

Sethi Ullman: Labeling Done

Val: '1' Label: 1    Val: '+' Label: 1    Val: '2' Label: 0    Val: '*' Label: 2    Val: '3' Label: 1    Val: '+' Label: 1    Val: '4' Label: 0

Sethi Ullman: CodeGen

Number of Available Registers: 2


MOV 1 , R1
OP+ 2 , R1
MOV 3 , R2
OP+ 4 , R2
OP* R2 , R1


======End======

By: Ravi Jain (411764) (CSE)
```

4.



```
(base) ravijain@ravijain-HP-Pavilion-Laptop-15-cc1xx:~/Documents/Sublime Text/LP LAB/Sethi Ullman$ ./a.out
Input: (((1)*(2+3))*(4+5))

Expression Tree Made

Root Top: *

Sethi Ullman: Labeling Done

Val: '1' Label: 1    Val: '*' Label: 2    Val: '2' Label: 1    Val: '+' Label: 1    Val: '3' Label: 0    Val: '*' Label: 2    Val: '4' Label: 1    Val: '+' Label: 1    Val: '5' Label: 0

Sethi Ullman: CodeGen

Number of Available Registers: 2


MOV 1 , R1
MOV 2 , R2
OP+ 3 , R2
OP* R2 , R1
MOV 4 , R2
OP+ 5 , R2
OP* R2 , R1


======End======

By: Ravi Jain (411764) (CSE)
```

5.



```
(base) ravijain@ravijain-HP-Pavilion-Laptop-15-cc1xx:~/Documents/Sublime Text/LP LAB/Sethi Ullman$ ./a.out
Input: ((((1+2)*(3+4))*(5-6))/(7+8))

Expression Tree Made

Root Top: /

Sethi Ullman: Labeling Done

Val: '1' Label: 1    Val: '+' Label: 1    Val: '2' Label: 0    Val: '*' Label: 2    Val: '3' Label: 1    Val: '+' Label: 1    Val: '4' Label: 0    Val: '*' Label: 2    Val: '5' Label: 1    Val: '-' Label:
 1    Val: '6' Label: 0    Val: '/' Label: 2    Val: '7' Label: 1    Val: '+' Label: 1    Val: '8' Label: 0

Sethi Ullman: CodeGen

Number of Available Registers: 2


MOV 1 , R1
OP+ 2 , R1
MOV 3 , R2
OP+ 4 , R2
OP* R2 , R1
MOV 5 , R2
OP- 6 , R2
OP* R2 , R1
MOV 7 , R2
OP+ 8 , R2
OP/ R2 , R1


======End======

By: Ravi Jain (411764) (CSE)
```