

**NAME- ABHIJIT BARIK REG. NO:**

ASSIGNMENT-1

1.

.Students of a Programming class arrives to submit assignments. Their register numbers are stored in a LIFO list in the order in which the assignments are submitted. Write a program using array to display the register number of the ten students who submitted first.

Register number of the ten students who submitted first will be at the bottom of the LIFO list. Hence pop out the required number of elements from the top so as to retrieve and display the first 10 students.

Low Level: Display the register number of the last submitted record [6 Marks]

Middle Level: Display the register number of the ten students who submitted first [2 Marks]

High Level: Any query posed by faculty such as checking if a particular student has submitted the assignment or not [2 Marks]

**CODE-(HIGH LEVEL,MID LEVEL,LOW LEVEL)**

#include <iostream>

using namespace std;

#define SIZE 30

void push(string);

void pop();

void display();

void peek();

void submission(string);

void firstten();

string stack[SIZE];

int top = -1;

int main(){

int choice;

string v,s;

int f=1;

while(1){

cout<<"\n"<<"\*\*\*\*\* MENU \*\*\*\*\*"<<"\n"<<endl;

cout<<"1. Push\n2. Pop\n3. Display\n4. Peek Last Submission\n5. Search For submission\n6. First 10 students submission\n7. Exit"<<endl;

cout<<"\nEnter your choice:" <<endl;

cin>>choice;

switch(choice){

case 1:

cout<<"Enter the registration no. of student:";

cin>>v;

//To avoid repetation of same data entry

if(top!=-1){

int t=top;

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

if(v==stack[j]){

f=0;

cout<<"\nNumber already Exists\n";

break;

}

}

if(f==0) continue;

push(v);

}

else{

push(v);

cin.ignore();

}

break;

case 2:

pop();

break;

case 3:

cout<<"Element of stack: ";

display();

break;

case 4:

cout<<"Last register number of the last submitted record"<<endl;

peek();

break;

case 5:

cout<<"Enter the Register number of a student to check they are submitted assignment or not:"<<endl;

cin>>s;

submission(s);

break;

case 6:

cout<<"Register number of the ten students who submitted first"<<endl;

firstten();

break;

case 7: exit(0);

default: cout<<"\nWrong selection!!! Try again!!!"<<endl;

}

}

return 0;

}

void push(string value){

if(top==SIZE-1){

cout<<"stack is full"<<endl;

}

else{

top++;

stack[top]= value;

cout<<"insertion sucessfull"<<endl;

}

}

void pop(){

if(top == -1)

{

cout<<"\nStack is Empty!!! Deletion is not possible!!!"<<endl;

}

else{

cout<<"\nDeleted :"<< stack[top]<<endl;

top--;

}

}

void display(){

if(top==-1)

cout<<"stack is empty!!!"<<endl;

else{

for(int i=top;i>=0;i--){

cout<<stack[i]<<"---" ;

}

}

}

void peek(){

if(top==-1){

cout<<"stack is empty"<<endl;

}

else

cout<<stack[top]<<endl;

}

void submission( string x){

int f=0;

if(top==-1)

cout<<"stack is empty!!!"<<endl;

for(int i=top;i>=0;i--){

if(stack[i]==x)

{

f=1;

break;

}

}

if(f==1)

cout<<" SUBMITTED"<<endl;

else

cout<<"NOT SUBMITTED YET"<<endl;

}

void firstten(){

while(top>=10){

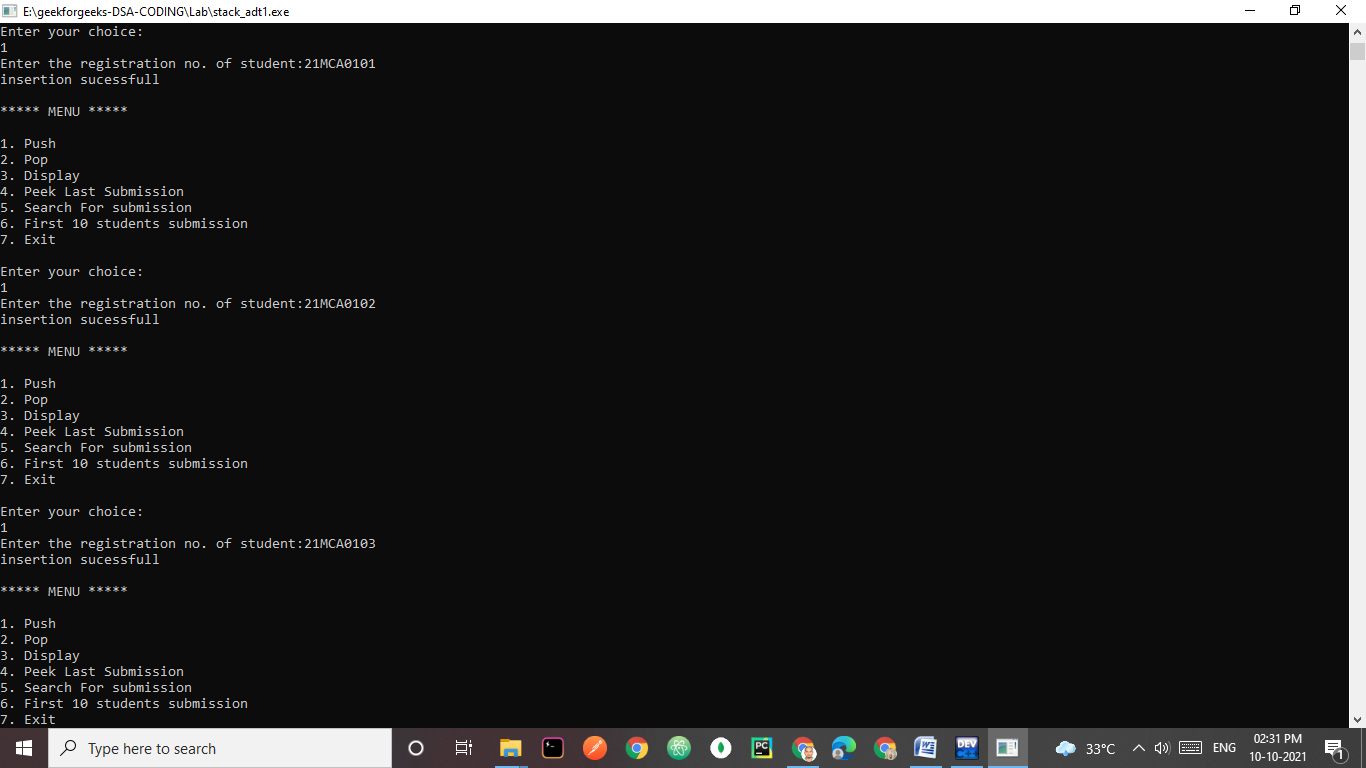
pop();

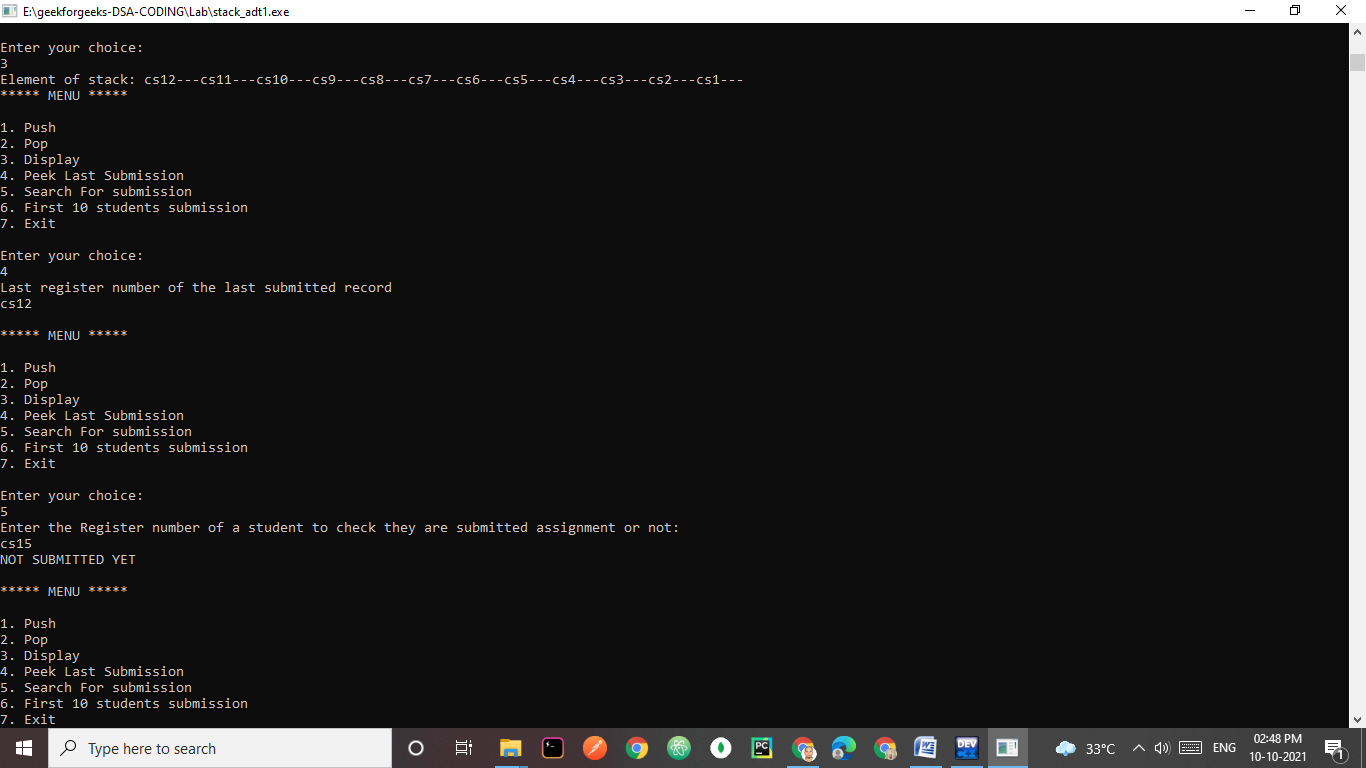
}

display();

}

**OUTPUT:-**





ASSIGNMENT-2

2.

To facilitate a thorough net surfing, any web browser has back and forward buttons that allow the user to move backward and forward through a series of web pages. To allow the user to move both forward and backward two stacks are employed. When the user presses the back button, the link to the current web page is stored on a separate stack for the forward button. As the user moves backward through a series of previous pages, the link to each page is moved in turn from the back to the forward stack.

When the user presses the forward button, the action is the reverse of the back button. Now the item from the forward stack is popped, and becomes the current web page. The previous web page is pushed on the back stack. Simulate the functioning of these buttons using array implementation of

Stack. Also provide options for displaying the contents of both the stacks whenever required.

Low Level: Implement either forward stack or backward stack [6 Marks]

Middle Level: Implement web browser navigation using both the stacks [2 Marks]

High Level: Use a single array to implement both the stacks [2 Marks]

**CODE-(HIGH LEVEL,MID LEVEL,LOW LEVEL)**

#include <iostream>

using namespace std;

#define SIZE 20

string backward\_stack[SIZE];

string forward\_stack[SIZE];

int top\_backward = -1;

int top\_forward=-1;

string current\_add;// global variable

void push(string current\_add); //push.pop,display function for low level .backward stack

void pop();

void display();

int isfull();

int isempty();

//this mainly for forward stack.midlevel

int isempty\_forward();

int isfull\_forward();

void pop\_forward();

void display\_forward();

void push\_forward(string x);

//this is for higlevel

string z; //global variable

string stack[SIZE];

int top\_back=-1;

int top\_for=SIZE;

void push\_back(string x){

if(top\_back<top\_for-1){

top\_back++;

stack[top\_back]=x;

}

else{

cout<<"stack is full"<<endl;

}

}

void push\_for(string x){

if(top\_back<top\_for-1){

top\_for--;

stack[top\_for]=x;

}

else{

cout<<"stack is full"<<endl;

}

}

void pop\_back(){

if(top\_back>=0){

z=stack[top\_back];

top\_back--;

}

else{

cout<<"stack is empty";

}

}

void pop\_for(){

if(top\_for<SIZE){

z=stack[top\_for];

top\_for++;

}

else{

cout<<"stack is full";

}

}

void displaystack()

{

int i=0;

cout<<"The current page URL is:"<<z<<endl;

cout<<"FORWARD PAGES LIST:\n";

if(top\_for==SIZE)

cout<<"NO WEBPAGES\n";

else

{

for(i=top\_for;i<SIZE;i++)

cout<<stack[i]<<endl;

}

cout<<"BACKWARD PAGES LIST:\n"<<endl;

if(top\_back==-1)

cout<<"NO WEBPAGES\n";

else{

for(i=top\_back;i>=0;i--)

cout<<stack[i]<<endl;

}

}

int main(){

int choice;

string x;//taking input web address in this variable low level

int first=0,innerlevel\_choice=1;

while(choice!=4){

cout<<"\n"<<"\*\*\*\*\* MENU \*\*\*\*\*"<<"\n"<<endl;

cout<<"1. Low Level\n2. Mid Level\n3. High Level\n4. Exit"<<endl;

cout<<"\nEnter your choice level:" <<endl;

cin>>choice;

switch(choice){

case 1:while(innerlevel\_choice!=3){

cout<<"1.Go to a new web address\n";

cout<<"2.Go back to previous web address(use after adding new web address)\n";

cout<<"3.Exit\n";

cout<<"Enter your choice from low level\n";

cin>>innerlevel\_choice;

switch(innerlevel\_choice)

{

case 1:

cout<<"Enter the web address\n";

cin>>x;

push(current\_add);

current\_add=x;

display();

cout<<"The current site is "<<current\_add<<endl;

break;

case 2: current\_add=backward\_stack[top\_backward];

pop();

display();

cout<<"The current site is "<<current\_add<<endl;

break;

}

}

break;

case 2:while(innerlevel\_choice!=4){

cout<<"1.Go ahead(use after Go Back)\n";

cout<<"2.Go back(Dont use in empty backward stack)\n";

cout<<"3.Go to a new address\n";

cout<<"4.Exit\n";

cout<<"Enter your choice from mid level \n";

cin>>innerlevel\_choice;

switch(innerlevel\_choice){

case 1: push(current\_add);//pusing in backward stack

current\_add=forward\_stack[top\_forward];

pop\_forward();

display(); //backward stack displaying

cout<<"The current site is "<<current\_add<<endl;

display\_forward();

break;

case 2: push\_forward(current\_add);

current\_add=backward\_stack[top\_backward];

pop();

display();//display backward stack

cout<<"The current site is "<<current\_add<<endl;

display\_forward();

break;

case 3: cout<<"Enter the web address\n";

cin>>x;

if(first>0){

push(current\_add);

}

first++;

current\_add=x;

display();

cout<<"The current site is "<<current\_add<<endl;;

top\_forward= -1;

display\_forward();

break;

}

}

break;

case 3:

int t = 0;

cout<<"Enter the new URL"<<endl;

cin>>z;

cout<<"\n";

while(1)

{

cout<<"\nOPTIONS:\n1. Forward\n2. Backward(use after Forward& dont use it if Backward stack is Empty )\n3. Stop\n";

cout<<"Enter your choice from High Level: ";

cin>>t;

switch(t){

case 1:

push\_back(z);

if(top\_for==SIZE){

cout<<"Enter the new URL"<<endl;

cin>>z;

}

else

pop\_for();

displaystack();

break;

case 2:

push\_for(z);

pop\_back();

displaystack();

break;

case 3:

cout<<"END"<<endl;

exit(0);

}

}

break;

// default: cout<<"\nWrong selection!!! Try again!!!"<<endl;

}

}

return 0;

}

//this for mainly low level//

int isempty(){

if(top\_backward==-1)

return 1;

else

return 0;

}

int isfull(){

if(top\_backward==SIZE-1)

return 1;

else

return 0;

}

void push(string current\_add){

if(isfull())

cout<<"stack is full\n";

else{

top\_backward++;

backward\_stack[top\_backward]=current\_add;

}

}

void pop(){

if(isempty()){

cout<<"stack is empty\n";

}

else{

top\_backward--;

}

}

void display(){

int i;

cout<<"web pages in backward stack\n";

if(isempty())

cout<<"backward stack is empty!\n";

else{

for(i=top\_backward;i>=0;i--)

cout<<backward\_stack[i]<<endl;

}

}

// this for mainly mid level

int isempty\_forward(){

if(top\_forward==-1)

return 1;

else

return 0;

}

int isfull\_forward(){

if(top\_forward==SIZE-1)

return 1;

else

return 0;

}

void push\_forward(string current\_add){

int i;

if(isfull\_forward())

cout<<"stack is full\n"<<endl;

else{

top\_forward++;

forward\_stack[top\_forward]=current\_add;

}

}

void pop\_forward(){

if(isempty\_forward()){

cout<<"stack is empty\n";

}

else{

top\_forward--;

}

}

void display\_forward(){

int i;

cout<<"web adress in forward stack\n";

if(isempty\_forward())

cout<<" forward stack is empty\n";

else{

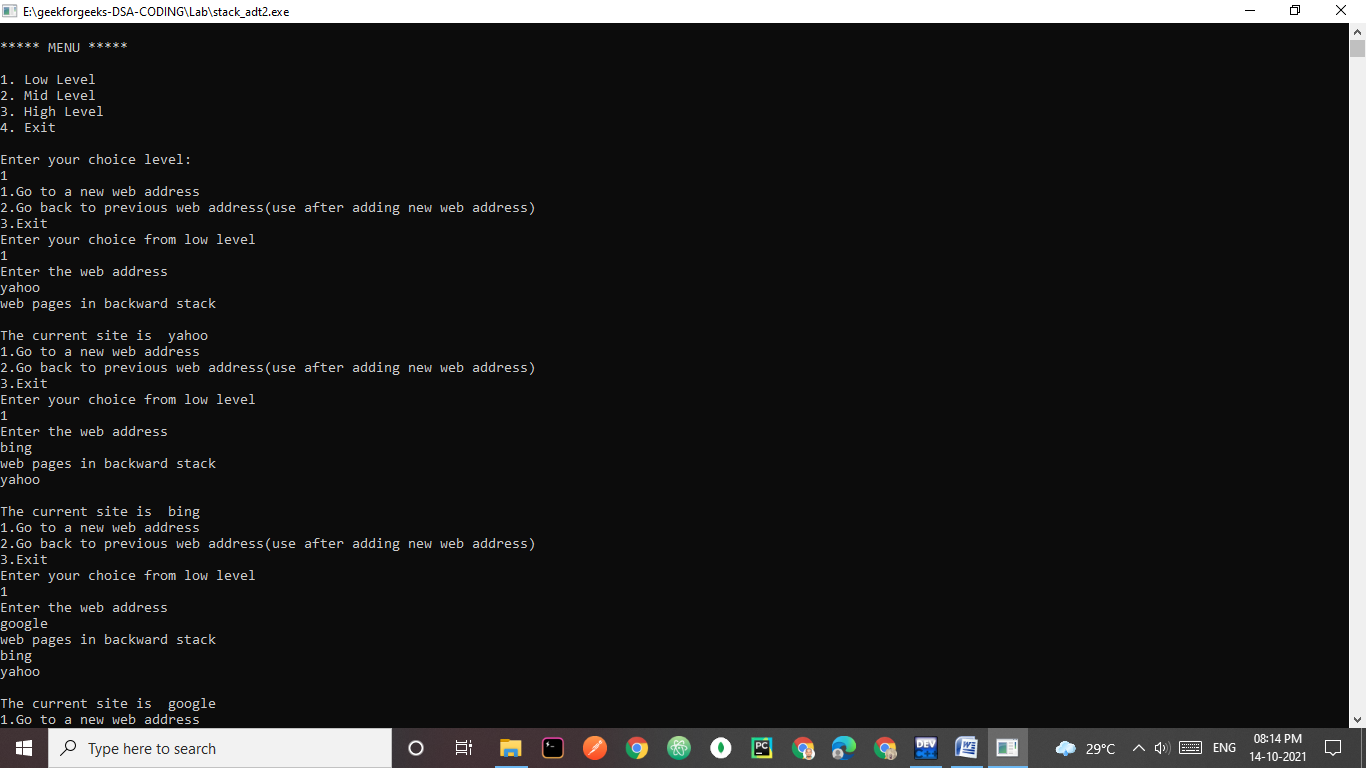
for(i=top\_forward;i>=0;i--)

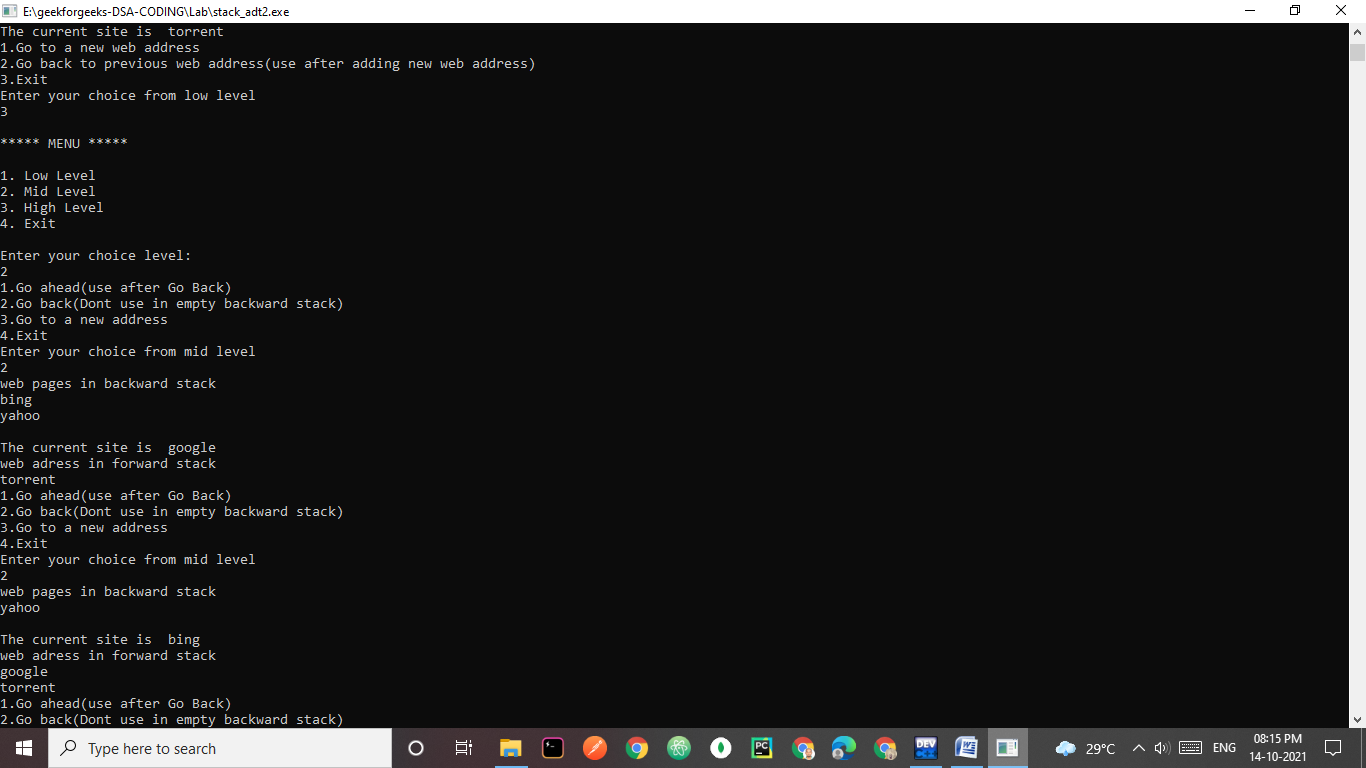
cout<<forward\_stack[i]<<endl;

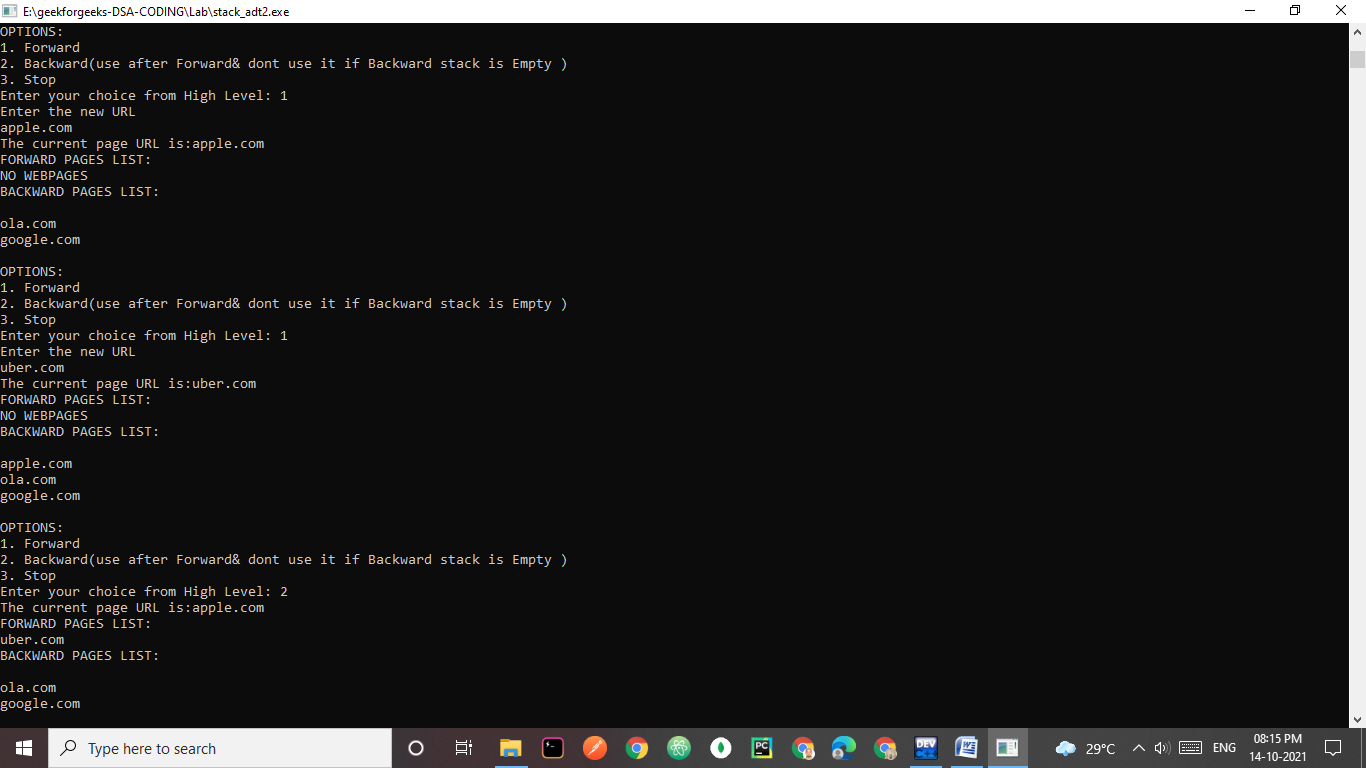
}

}

**OUTPUT:-**







ASSIGNMENT-3

3.Most of the bugs in scientific and engineering applications are due to improper usage of precedence order in arithmetic expressions. Thus it is necessary to use an appropriate notation that would evaluate the expression without taking into account the precedence order and parenthesis.

a) Write a program to convert the given arithmetic expression into

i) Reverse Polish notation

ii) Polish notation

b) Evaluate the above notations with necessary input.

Low Level: Implement one conversion and its evaluation [6 marks]

Middle Level: Implement both polish and reverse polish conversions [2 marks]

High Level: Develop a menu driven program to implement all conversions and evaluations in a single program with different inputs [2 marks]

**CODE:-(HIGH LEVEL,MID LEVEL,LOW LEVEL)**

#include <bits/stdc++.h>

using namespace std;

//infixToPostfix

#define SIZE 20

void push(char);

void pop();

char peek();

int top=-1;

char st[SIZE];

int prec(char c)

{

if(c == '^')

return 3;

else if(c == '\*' || c == '/')

return 2;

else if(c == '+' || c == '-')

return 1;

else

return -1;

}

// The main function to convert infix

// expression to postfix expression

string infixToPostfix(string s)

{

//char st[50];

push('@');

int l = s.length();

string ns;

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

{

// If the scanned character is an operand,

// add it to output string.

if((s[i] >= 'a' && s[i] <= 'z')||(s[i] >= 'A' && s[i] <= 'Z')||(s[i] >= '0' && s[i] <= '9'))

ns+=s[i];

else if(s[i] == '(')

push('(');

else if(s[i] == ')')

{

while(peek() != '@' && peek() != '(')

{

char c = peek();

pop();

ns += c;

}

if(peek() == '(')

{

char c = peek();

pop();

}

}

// If an operator is scanned

else{

while(peek() != '@' && prec(s[i]) <= prec(peek()))

{

char c = peek();

pop();

ns += c;

}

push(s[i]);

}

}

// Pop all the remaining elements from the stack

while(peek() != '@')

{

char c = peek();

pop();

ns += c;

}

return ns;

//cout <<"Postfix Expression:" <<ns << endl;

}

string infixToprefix(string s ){

push('@');

string result;

int n=s.length();

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

if((s[i]>='a'&& s[i]<='z')|| (s[i]>='A'&& s[i]<='Z')||(s[i]>='0'&& s[i]<='9'))

{

result+=s[i];

}

else if(s[i]==')')

push(s[i]);

else if(s[i]=='('){

while(peek()!='@' && peek()!=')'){

char c = peek();

pop();

result+=c;

}

if(peek() == ')')

{

char c=peek();

pop();

}

}

else if(s[i]=='^'){

while(peek()!='@'&& prec(s[i])<=prec(peek())){

char c=peek();

pop();

result+=c;

}

push(s[i]);

}

else{

while(peek()!='@'&& prec(s[i])<prec(peek())){

char c=peek();

pop();

result+=c;

}

push(s[i]);

}

}

while(peek() != '@')

{

char c = peek();

pop();

result += c;

}

result=string(result.rbegin(), result.rend());

return result;

}

//evaluation

int sti[SIZE];

int peeki(){

if(top==-1){

cout<<"stack is empty"<<endl;

}

else

return sti[top];

}

void pushi(int value){

if(top==SIZE-1){

cout<<"stack is full"<<endl;

}

else{

top++;

sti[top]= value;

}

}

int evaluatePostfix(string exp)

{

// Create a stack

int i;

// Scan all characters one by one

for (i = 0; exp[i]; ++i)

{

// If the scanned character is an

// operand (number here), extract the full number

// Push it to the stack.

if (isdigit(exp[i]))

{

pushi(exp[i] - '0');

}

// If the scanned character is an operator, pop two

// elements from stack apply the operator

else

{

int val1 = peeki();

pop();

int val2 = peeki();

pop();

switch (exp[i])

{

case '+': pushi(val2+val1); break;

case '-':pushi(val2-val1); break;

case '\*':pushi(val2\*val1); break;

case '/':pushi(val2/val1); break;

}

}

}

return peeki();

}

int evaluatePrefix(string exp)

{

int i;

// Scan all characters one by one

for (i=exp.size()-1;i>=0;--i)

{

if (isdigit(exp[i]))

{

pushi(exp[i] - '0');

}

// If the scanned character is an operator, pop two

// elements from stack apply the operator

else

{

int val1 = peeki();

pop();

int val2 = peeki();

pop();

switch (exp[i])

{

case '+': pushi(val1+val2); break;

case '-':pushi(val1-val2); break;

case '\*':pushi(val1\*val2); break;

case '/':pushi(val1/val2); break;

}

}

}

return peeki();

}

int main(){

int choice;

string exp,ns;

while(1){

cout<<"\n"<<"\*\*\*\*\*HIGH Level MENU \*\*\*\*\*"<<"\n"<<endl;

cout<<"1. POSTFIX conversion\n2. PREFIX conversion\n3. POSTFIX CONVERSION AND EVALUATION\n4. PREFIX CONVERSION AND EVALUATION\n5. Exit"<<endl;

cout<<"Enter your choice:\n";

cin>>choice;

switch(choice){

case 1:

//string exp;

cout<<"ENTER THE INFIX STRING TO CONVERT TO POSTFIX"<<endl;

cin>>exp;

cout <<"Postfix Expression:" << infixToPostfix(exp) << endl;

break;

case 2:

//string exp;

cout<<"ENTER THE INFIX STRING TO CONVERT TO PREFIX"<<endl;

cin>>exp;

infixToprefix(exp);

cout <<"Prefix Expression:" << infixToprefix(exp) << endl;

break;

case 3:

cout<<"ENTER THE INFIX STRING FOR POSTFIX CONVERSION AND EVALUATION"<<endl;

cin>>exp;

ns=infixToPostfix(exp);

cout<<"POSTFIX EXPRESSION:"<<ns<<endl;

cout <<"POSTFIX EVALUATION OF THIS INFIX"<<ns<<" IS: "<<evaluatePostfix(ns);

break;

case 4:

cout<<"ENTER THE INFIX STRING FOR PRETFIX CONVERSION AND EVALUATION"<<endl;

cin>>exp;

ns=infixToprefix(exp);

cout<<"PREFIX EXPRESSION:"<<ns<<endl;

cout <<"PREFIX EVALUATION OF THIS INFIX"<<ns<<" IS: "<<evaluatePrefix(ns);

break;

case 5:

exit(0);

}

}

return 0;

}

void push(char value){

if(top==SIZE-1)

cout<<"pstack is full"<<endl;

else{

top++;

st[top]=value;

}

}

void pop(){

if(top==-1)

cout<<"popstack is empty"<<endl;

else

top--;

}

char peek(){

if(top==-1)

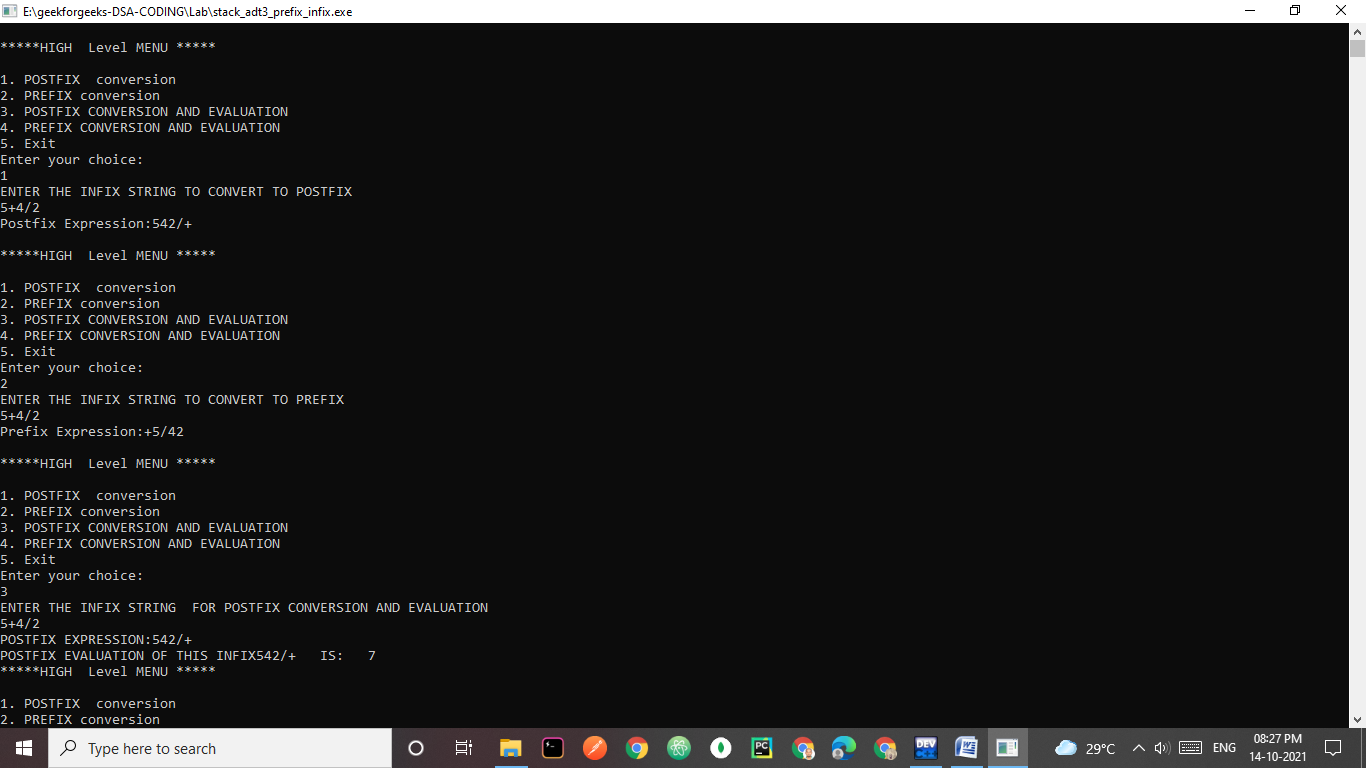
cout<<"peekstack is empty"<<endl;

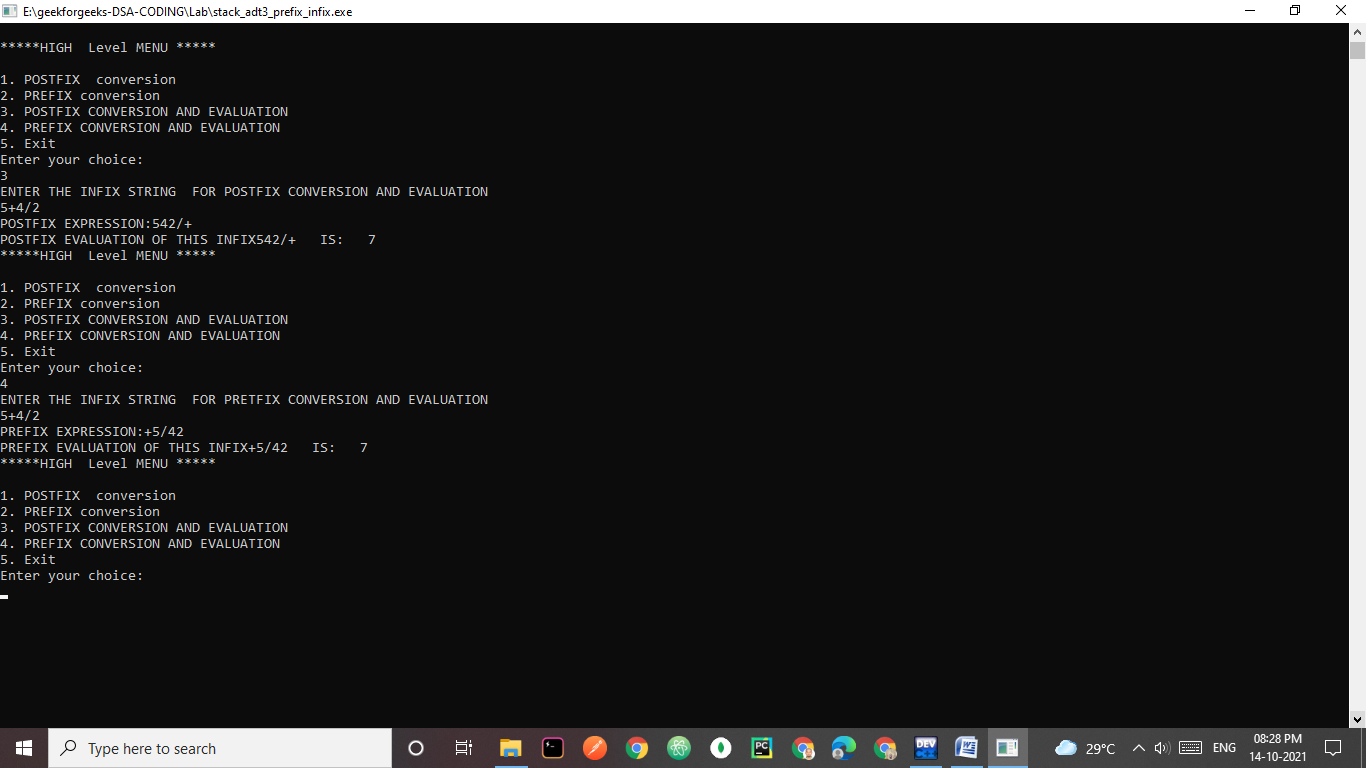
else

return st[top];

}

**OUTPUT:-**

****

****