**Multithreading Basics and Synchronization Mechanisms Assignments**

**Mandatory Assignments**

1.WAP to

a. read a list of email id's separated by ; as a single string command line argument

b. Extract each email id , pass it as an argument to a thread for validation

c. Create one thread/email id to process.

d. Each thread to perform following validations to ensure valid email id.

i. user name should begin with an alphabet

ii. domain name should be “.com" or ".edu"

e. On valid email id, it should increment global variable named "validmail\_count".

f. On valid email id, return the extracted valid username to main thread, else return NULL

Ans:

1 #include<iostream>

2 #include<string>

3 #include<cstring>

4 #include<thread>

5 #include<vector>

6 #include<ostream>

7 #include<sstream>

8 using namespace std;

9 void validate(string s){

10 if(isalpha(s[0])){

11 string s1=s.substr(s.length()-4);

12 if(s1==".com" || s1==".edu"){

13 cout <<s<<" - Valid"<<endl;

14 }

15 else{

16 cout <<s<<" - Not Valid"<<endl;

17 }

18 }

19 else{

20 cout <<s<<" -Not Valid"<<endl;

21 }

22 }

23 int main(int argc, char\* argv[])

24 {

25 int i;

26 char \*ptr;

27 //const char \*s1=",";

28 //thread t1(validate);

29 string str;

30 str=argv[1];

31 string s[100],s1,T;

32 stringstream X(str);

33 int j=0;

34 int m=0;

35 if(str.length()>1){

36 j=0;

37 while(getline(X,T,',')){

38 s[j++]=T;

39 m+=1;

40 }

41 }

42 // cout<<str<<"\n"<<argv[1]<<"\n";

43 // cout << s[1] <<"\n";

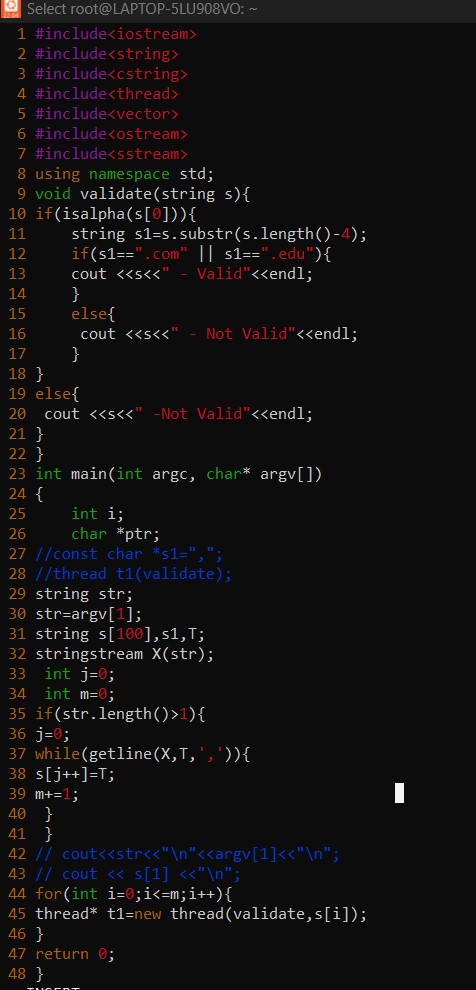
44 for(int i=0;i<=m;i++){

45 thread\* t1=new thread(validate,s[i]);

46 }

47 return 0;

48 }



1. Write application demonstrating thread creation using function callback, function object, lambda functions

Ans:

#include <iostream>

2 #include <thread>

3 using namespace std;

4 class thr

5 {

6 private:

7 int n;

8 public:

9 int operator()(int x)

10 {

11 cout<<"Function call"<<endl;

12 cout <<x<<endl;

13 return 0;

14 }

15 };

16 void threadfn(int value)

17 {

18 cout<<"hello"<<endl;

19 cout<<"value:"<<value<<endl;

20 }

21 int main()

22 {

23 thr a;

24 //std::thread t2(thr::threadfun,5)

25 int lvalue=100;

26 thread t1(threadfn,lvalue);//calling function

27 thread t2(thr(),5);//calling object

28 thread t3{[](int &value){

29 cout<<"value in t3:"<<value++<<endl;},ref(lvalue)};//Lamda function

30 t1.join();

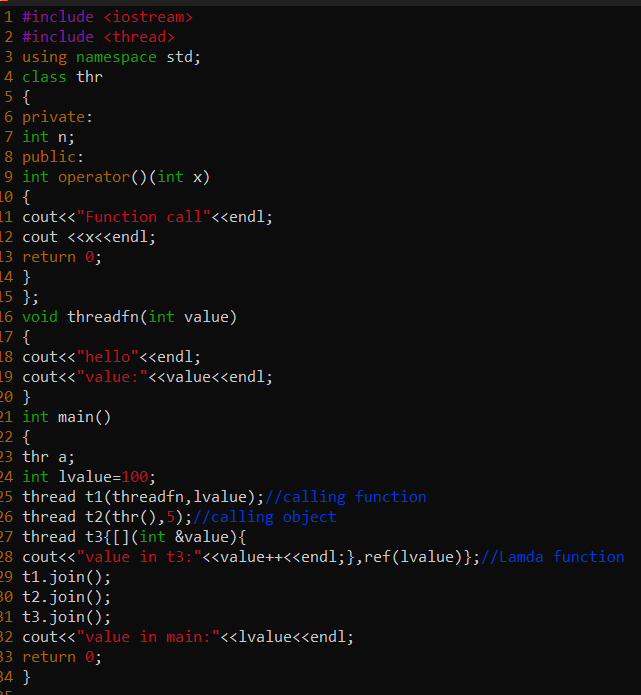
31 t2.join();

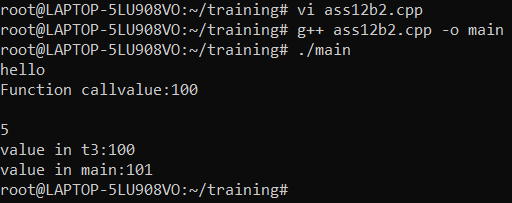
32 t3.join();

33 cout<<"value in main:"<<lvalue<<endl;

34 return 0;

35 }





3. WAP to create 2 threads, each one to be passed with a line of text to be used as input, then read 2 substrings to be searched for from the user. Pass one substring each to thread using promise set\_value(). Each thread should search for one or more occurrence of the substring and return the number of occurences to caller. Caller to use get() and read the value.