**Multithreading Basics and Synchronization Mechanisms Assignments**

**Mandatory Assignments**

1. WAP to
   1. 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 nameshould 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.





#include<iostream>

2 #include<string>

3 #include<cstring>

4 #include<thread>

5 #include<ostream>

6 #include<vector>

7 #include<sstream>

8

9 using namespace std;

10

11 void verify(string s){

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

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

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

15 cout<<s<< "- valid" <<endl;

16 }

17 else{

18 cout<<s<<"-invalid"<<endl;

19 }

20 }

21 else{

22 cout<<s<<"-invalid"<<endl;

23 }

24 }

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

26 {

27 int i;

28 char \*ptr;

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

43 thread\* t1=new thread(verify,s[i]);

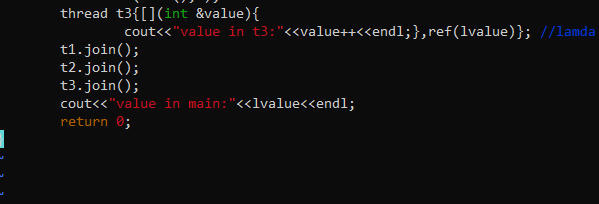
44 }

45

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

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.





1 #include<iostream>

2 #include<thread>

3

4 using namespace std;

5 class Thre

6 {

7 private:

8 int n;

9 public:

10 int operator()(int x)

11 {

12 cout<<"Function call"<<endl;

13 cout<<x<<endl;

14 return 0;

15 }

16 };

17 void threadfunc(int value)

18 {

19 cout<<"this is"<<endl;

20 cout<<"value:"<<value<<endl;

21 }

22 int main()

23 {

24 Thre t;

25 int lvalue=50;

26 thread t1(threadfunc,lvalue);

27 thread t2(Thre(),5);

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

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

30 t1.join();

31 t2.join();

32 t3.join();

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

34 return 0;

35 }

