Multithreading Basics and Synchronization Mechanisms 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.

Answer:

```
#include<iostream>
#include<string>
#include<cstring>
#include<thread>
#include<vector>
#include<ostream>
#include<sstream>

using namespace std;

 void validate(string s){
         if(isalpha(s[0])){

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

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

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

                 }

                 else{

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

                 }

          }
            else{

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

            }

 }

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

{

        int i;
        cout<<"Enter the arguments "<< argc <<endl;
```

```
                {
                        cout<<argv[i]<<endl;
                }
        char *ptr;
        string str;
        str=argv[1];
        string s[100],s1,T;
        stringstream X(str);
        int j=0;
        int m=0;
        if(str.length()>1){
                j=0;
                while(getline(X,T,',')){
                        s[j++]=T;
                        m+=1;
                }
        }

        for(int i=0;i<=m;i++){
                thread* t1=new thread(validate,s[i]);
        }
        return 0;
}
```

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

   thread created using function object

   Answer:

```
// thread created using function object
 #include <iostream>
 #include <thread>

 using namespace std;
 class Person
 {
         public:
         void operator()(){
                 for(int i=1;i<=10;i++)
                         cout<<17*i<<endl;
                         }
};
int main()
{
         thread t1((Person()));
         cout<<"\nExecuting main thread"<<endl;
         for(int i=0;i<20;i++)
         cout<<"Main Thread"<<i<<" = "<<i<<endl;
         cout<<"\nExecuting the thread"<<endl;
         t1.join();

         return 0;


}
```

Thread created by using function calling

```cpp
//Thread created by function calling
#include <iostream>
#include <thread>
using namespace std;
void Function()
{
        for(int i=1;i<=10;i++)
        cout<<10*i<<endl;
}
int main()
{
        thread t1(Function);
        for(int i=0;i<30;i++)
        cout<<"Thread"<<i<<" = "<<i<<endl;
        t1.join();
        t1.detach();
        return 0;
}
```

Thread created by lambda function

```cpp
//Thread using lamda function
#include <iostream>
#include<thread>

using namespace std;

int main()

{

        std::thread t1([]{
        for(int i=1;i<=5;i++)
        cout<<9*i<<endl;
        });

        cout<<"\nExecuting main thread"<<endl;
        for(int i=0;i<25;i++)
        cout<<"in main i = "<<i<<endl;
        cout<<"\nExecuting the thread"<<endl;
        t1.join();
        return 0;
}
```

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.

Answer:

```
#include <future>
#include <iostream>
#include <thread>
#include <utility>
using namespace std;

void product(promise<string>&& intPromise, string a, string b){
          intPromise.set_value(a);
}

struct Div{

        void operator() (promise<string>&& intPromise, string a, string b) const {
        intPromise.set_value(b);
        }

};

int main(){

        string a= "Capgemini";
        string b= "Company";

        cout << endl;


        promise<string> prodPromise;
        promise<string> divPromise;


        future<string> prodResult= prodPromise.get_future();
        future<string> divResult= divPromise.get_future();


        thread prodThread(product,move(prodPromise),a,b);
        Div div;
        thread divThread(div,move(divPromise),a,b);


        cout << prodResult.get() << endl;
        cout << divResult.get() << endl;

        prodThread.join();

        divThread.join();

        cout <<endl;

}
-- INSERT --
```