Name: Baber Khalid
University: Lahore University of Management Sciences
Major: Computer Science
Degree Programme: B.Sc
Email: 18100203@lums.edu.pk
Availability:

- I plan to spend 20-22 Hrs/Week on GSOC.
- I plan to start from May 25, 2017.
- There are no factors which can affect my availability for three months of summer, currently.

Background Info:

I have currently completely my Intermediate degree which means 12 years of study. During my current program, I have taken **Introduction to Programming, Data Structures, Algorithms, Intro to Artificial Intelligence, Operating Systems, Databases, Network Centric Computing, Fundamentals of Computer Systems, Computer Graphics** and I am currently enrolled in **Advanced Programming, Advanced Operating Systems, Software Engineering,** and **Network Security.**

Some of the projects I have worked on before include:

1. Did several tasks using the toy Operating System **Nachos** most fun of which writing code for **managing virtual address space** to enable multi processing on it. This was done in **C++**.
2. Worked with an instructor of Computer Science department in a **research internship** related to the area **of Networks Censorship**, in which the aim was to minimize latency by optimizing path selection for **TOR** using different techniques like **user profiling** and **path selection depending on geographical distances**.
3. Wrote a **cache simulator** in **C** employing least recently used policy for replacement**.**
4. Wrote a **multi-threaded server in C using pthreads** and **socket programming.**
5. Developed a **Dropbox like application** which automatically synced client and server directories **in Java.**
6. Have basic experience of using **llvm** and **klee**.

I like to solve problems which seem fun using programming or things which introduce me to completely new concepts.

When I was searching for an organization to apply for summer of code, I found out about Boost. I knew about it before but I had never used any boost libraries. I talked about boost with my advisor and he told me about template metaprogramming and how Boost has several libraries which optimize code at the compile time. The idea was fascinating that how a compiler could do things at compile time. This became my reason for trying to contribute towards Boost libraries.

I checked out your ideas and the very first idea which proposed the **problem of writing a static_map** which did everything including hashing on the compile time seemed interesting to me. I did some research on the different techniques used to make the compiler execute code at compile time including template metaprogramming and dived into writing the code for

competency test. And this is my motivation for applying to Boost for Google Summer of Code 2017.

Writing the code for competency test was the very **first time**, I experienced working in this area of code.

After and during this summer of code, I plan on learning similar techniques as this world is programming seems different and exciting to me.

My expertise in the following is:

1. **C++98/03** => 3 out of 5
2. **C++11/14** => 2 out of 5
3. **C++ Standard Library** => 3 out of 5
4. **Boost Libraries** => 0 out of 5
5. **Git** => 3 out of 5

I am most familiar with Visual Studio but prefer to work using a text editor.
Even though I have never written any separate documentation for the code till now, but I have experience of reading documentation through doxygen and am willing to learn.

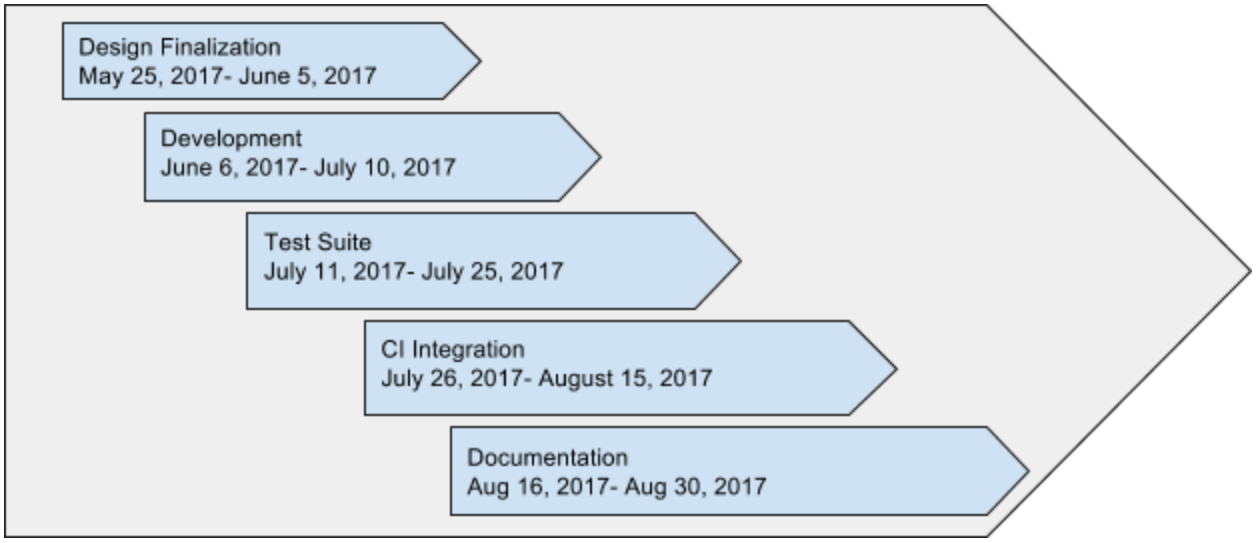
Proposed Project:

The project I propose is the development of a static_map class with a very simple design as follows:

1. Maintain an array of pair<true, pair<key, value>> inside a Map class.
2. For item lookups, use template metaprogramming to iterate over all the elements in the array if found return that data element , else throw an exception from the template function.

The reason for suggesting template metaprogramming for the lookups is because compiler resolves templates at compile time and the idea of using inline functions. Since in c++14 I can create temporary objects and can invoke methods in them, I can use recursive templates to invoke a one line function at each recursive step until I reach a base template. The **idea came into my mind from Haskell** in which I can write multiple one line definitions of a function using different function guards and I tried to employ this same technique while writing competency test code.

Project Milestones and Schedule:



Project Competency:

Since I was relatively new to the idea of compile time execution techniques, instead of directly diving in to write my own code, I decided to understand the provided code which was working with the clang. So I came across different and new topics like variadic templates, perfect forwarding etc. After understanding the code (I couldn't understand why we used pair<size_t, pair<key, value>> in the toy to store the elements) I did the following:

1. While using clang_static_map.cpp as sample started checking which techniques being used in this file were being optimized by gcc.
2. I noticed that at first call to [] operator, gcc always optimized the code but when I called it more than once it generated a significant amount of runtime code.
3. I made a hunch (which I am still not sure if it's true or not) that whenever I made the multiple calls to same functions gcc was fixing the address of the function and was using call instruction of assembly to call it at runtime.
4. I tried optimizing the lookups by using the same idea as the map was created but i found another idea more easy to implement which is discussed below.
5. This was the idea of using inline functions but the requirement of using them was that function had to be one line long.
6. I had experience of using Haskell before and the syntax of recursive templates seemed very similar to it as it let me write multiple one line definitions of same function using different template arguments, similar to function guards in Haskell.
7. Since I was still getting used to the syntax, I made the decision of using the given code as much as I could and to produce a piece of code which was working. Using this approach, results were favorable.
8. However, my code generated run-time code when the element i accessed was at the 0th index, because the code of operator() in the struct at<0> was evaluated and this function was not one line so compiler generated run-time code.

9. I tried using throw std::out_of_range() in a ternary operator in at<0>::() operator to take care of the multi-line function. It **did not generate any run-time code**, but compiler generated a **long warning** in this case which was **not pretty**.
10. In final version, I **solved the warning problem** by throwing the exception in a function which had its return type as data but only threw an exception.
11. Following is my code for the competency test:

```
#include <assert.h>
#include <string.h>
#include <stdlib.h>
#include <array>
#include <initializer_list>
#include <utility>
#include <iostream>
#include <string.h>

namespace bro{
   template <size_t i> struct at{
         template<class key, class data, size_t n> inline constexpr data
&operator()(std::pair<size_t, std::pair<const key, data>> (arr)[n], const key& ind) const {
               return ind == arr[i].second.first ? arr[i].second.second : at<i-1>()(arr, ind);
         }

         template<class key, class data, size_t n> inline constexpr const data &operator()(const
std::pair<size_t, std::pair<const key, data>> (&arr)[n], const key& ind) const{
               return ind == arr[i].second.first ? arr[i].second.second : at<i-1>()(arr, ind);
         }
   };

   template<> struct at<0>{
         template <class data> inline constexpr data &dummy(const data& x){
               throw std::out_of_range("key not in the map");
         }
         template <class data> inline constexpr const data &dummy(const data& x) const{
               throw std::out_of_range("key not in the map");
         }
         template<class key, class data, size_t n> inline constexpr data
&operator()(std::pair<size_t, std::pair<const key, data>> (arr)[n], const key& ind) const {
               return arr[0].second.first == ind ? arr[0].second.second :
dummy(arr[0].second.second);
               // if (arr[0].second.first == ind)
               //     return arr[0].second.second;
```

```
                // throw std::out_of_range("key not in map");
        }

        template<class key, class data, size_t n> inline constexpr const data &operator()(const
std::pair<size_t, std::pair<const key, data>> (&arr)[n], const key& ind) const {
                return arr[0].second.first == ind ? arr[0].second.second :
dummy(arr[0].second.second);
                // if (arr[0].second.first == ind)
                //     return arr[0].second.second;
                // throw std::out_of_range("key not in map");
        }
    };

}


// template<template<class key, class data> class... ele> constexpr data &actualCheck(ele...
arr){
// //     return;
// }

// template<class key, class data, size_t n, size_t... I> constexpr data
&findInMap(std::pair<size_t, std::pair<key, data>> (&arr)[n], std::index_sequence<I...>){
//     return actualCheck(arr[I]...);
// }

// template<class key, class data, size_t n> constexpr data &lookup(std::pair<size_t,
std::pair<key, data>> (&arr)[n]){
//     return findInMap(arr, std::make_index_sequence<n>());
// }

template<class key, class data, size_t no>
struct static_map{
using key_type = key;
using mapped_type = data;
using value_type = std::pair<const key_type, mapped_type>;

private:
    using _value_type = std::pair<size_t, value_type>;
    _value_type _values[no];

    static constexpr _value_type _new_value_type(const std::pair<key, data> &v)
    {
```

```
            return std::make_pair(0, std::make_pair(v.first, v.second));
    }
public:
    template<class... U> constexpr static_map(U &&...il) : _values{ _new_value_type(il)... } {}

    constexpr data& operator[](const key &ind) const{
            return const_cast<data &>(bro::at<no-1>()(_values, ind));
    }

    constexpr data& at(const key &ind) const{
            return const_cast<data &>(bro::at<no-1>()(_values, ind));
    }

};

constexpr std::pair<const size_t, const char*> map[] = {
  {5, "apple"},
  {8, "pear"}
};

template<class key, class data, size_t n, size_t... I> constexpr static_map<key, data, n>
make_map_from_array(const std::pair<const key, data> (&arr)[n], std::index_sequence<I...>){
    return static_map<key, data, n>(arr[I]...);
}

template<class key, class data, size_t n> constexpr static_map<key, data, n> make_map(const
std::pair<const key, data> (&arr)[n]){
    return make_map_from_array(arr, std::make_index_sequence<n>());
}

int main(){
    constexpr auto cmap = make_map(map);
    if (!cmap[5]) abort();
    const char* f = cmap[5];
    // std::cout << f << std::endl;
    return 0;
}
```