STUDENT INFORMATION SYSTEM

Presented By:

Srinivasan Vijayraghavan (IMT2016083)
Nishant Malpani (IMT2016095)
Vaibhav Kansagara (IMT2016068)
Raja Rakshit (IMT2016087)

The Vision:

Our project, 'The Student Information System', revolves around the fact that it can manage student's data with great ease. Our project employs a 'Menu-driven' system which makes it user-friendly. It also uses files, as database, to perform file handling operations such as add, search, delete and displaying records to manage students' records. The challenging feature of 'Search' is implemented using the "KMP (Knuth-Morris-Pratt) Substring Search" algorithm.

Ingredients of the students' data:

In this project, data from the students are collected keeping the far-reaching goal, for the user, at the back of our mind. The data collected has components such as:

- Individual's blood group
- Individual's hobbies/ interests
- Individual's grades

Our project even has the capability of tracking an individual's attendance, documenting student's health records and many other features. The above-mentioned features ensure the usage of this project in a positive and useful way.

Concepts used:

- Program Organization (in C)
- Arrays
- Pointers
- Pointers to Functions
- Dynamic Memory Allocation
- Menu-Driven System
- File-Handling features
- KMP Substring Search algorithm

The heart of the program:

KMP Substring Search

"KMP Substring Search" searches for a pattern in the given string. It is far more efficient from the linear search that someone would normally think of.

Ex: Say, we are searching for the substring 'abc' (Child String) in 'klhabcshd' (Parent String)

k l h a b c s h d (Parent String)

0 1 2 3 4 5 6 7 8

Upon close observation, we can clearly see that the substring ‘abc’ is starting from the 3rd position. But how would you tell a computer to do this? So, one of the methods is the linear search.

First, you will see if the 0th element of the parent string matches with the 0th element of the child string, since they don’t match, we’ll get to the next one. Even the 1st character of the parent string doesn’t match with the first character of the child string.

Upon repeating this process, we’ll eventually get to the 3rd element of the parent string where we find the there’s a character match. Then, we check for the other two elements also and we conclude that a match has been found. This method is quite doable but not at all efficient. When we take much larger parent strings, this method is not so efficient.

The KMP method makes use of previous match information and is thus much more efficient.

Ex: If we are searching for ‘abcdfabc’ (Child String) in ‘ghabcvkjabcdfabc’ (Parent String)

g h a b c d f a b a b c d f a b c

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

The first closest match occurs from position 2 (till 8). At position 9, it is a mismatch.

In the regular linear search, if there were a mismatch, we would just increment the position in the parent string and try to match it with the child string. But in KMP, we make use of the fact that an ‘ab’ has occurred once before. A suffix is a prefix in this case. We can directly try to match the 3rd element with the 9th element.

Goals:

- Implementing the KMP substring search.
- Enabling the user to search by name, roll number, hobbies, grades, etc.
- Enabling the user to add a new student record into the database by file handling.
- Enabling the user to edit a particular section of the database (i.e. a particular student) by file handling.