CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Generation of n digit OTP using linked list and n character CAPTCHA using linear queue.

OTP generation algorithms usually use build in function which gives pseudo randomness or randomness, creating prediction of successor. OTPs hackers difficult and also in the hash function, which can be used to generate a random number or value but hard to reverse and therefore difficult for an attacker or hackers to obtain the data or information that was used in the hash.

This is necessary as a verification part of result otherwise it might be simple to predict future OTPs by perceptive previous ones and use it in the future.

Concrete OTP algorithms vary greatly in their details.

Various approaches for the generation of OTPs area unit listed below:

Based on time-synchronization between the authentication server and therefore the consumer providing the generated outputs to the authentication (OTPs area unit are valid just for a brief amount of time)

Using the mathematical rule available to get a new random number supported the previous number and the new number generated is different from the previous generated number (OTPs area unit effectively a sequence and should be utilized in a predefined order).

Using a mathematical rule wherever the new numbers relies on a challenge (e.g., a random variety chosen from the availability of set of numbers by the authentication server or dealings details) and/or a counter. Although the pattern of the sequence of numbers repeats it can be ignored as the time period is very long.

OTP is mandatory for authorizing the following transactions

- Registration of the beneficiary bank accounts of other banks
- Bill payments
- Direct Debit (e-commerce)
- Quick Pay
- NEFT/RTGS transaction(for transaction value INR 10,000 and above)
- Funds transfer to other Deutsche Bank Accounts
- Mobile / e-mail ID change
- Alert profile change
- Demand draft issuance

CAPTCHA (Completely Automated Public Turning test to tell Computers and Human Apart) is a challenge response system designed to differentiate humans from robotic software programs Captcha ’s are used as security checks to determine the spammers and hackers from using forms on the web pages to insert malicious or frivolous code. The most common kind of captcha is that the text captcha which requires the user to view a distorted string of alphanumeric characters in an image and enter the characters in an attached form. Text captchas are also rendered as MP3 audio recording to meet the needs of the visually impaired. Just as with pictures bots can detect the presence of an audio file,but solely somebody's will listen and grasp the knowledge the file contains.Picture recognition captcha which are also commonly used, ask the users to identify the subset of images within a larger set of images. For instance, the user could also be given a collection of pictures and asked to click on all those that have cars in them.

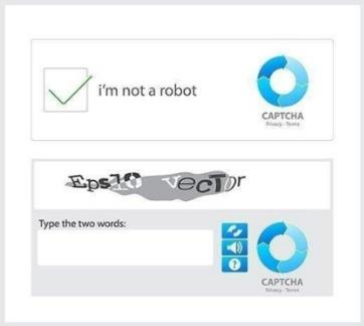
Other types of CAPTCHAs include:

o Math CAPTCHA - Requires the user to solve a basic math problem, such as adding or subtracting two numbers.

o 3D Super CAPTCHA - Requires the user to identify an image rendered in 3D.

o I am not a robot CAPTCHA - Requires the user to check a box

o Marketing CAPTCHA - Requires the user to type a particular word or phrase related to the sponsor's brand.



1.a CAPTCHA verification

1.2 OBJECTIVES

- Generation of both OTP and CAPTCHA.
- Generating n digit OTP using single linked list which contains random number varying from 0 to 9.
- Generating n characters CAPTCHA using linear queue which contains random characters varying from A to Z ,a to z and 0 to 9.
- Generating different output in each run or execution of the code.
- Current OTP output must be different from the previous OTP output.
- Current CAPTCHA output be different from previous outputs.
- The output is different even for same given n input.
- The Program must generate different output in different running of the program.

1.3 EXPECTED OUTCOMES

- Generation of required n digit OTP which has random numbers varying from 0 to 9 using linked list .
- Generation of required n characters CAPTCHA which has random characters varying from 0 to 9,a to z and A to Z using linear queue.
- Both OTP and CAPTCHA output should be different from previous outputs.
- The program must generate different output in different run or execution of program.

1.4 HARDWARE REQUIREMENTS

- **RAM** : 2GB or more
- **PROCESSOR** : Intel core 3i or more
- **INPUT :** Standard keyboard and mouse
- **OUTPUT :** Standard monitor

1.5 SOFTWARE REQUIREMENTS

- **COMPILERS** : Turbo C/C++ or any C compiler
- **OPERATING SYSTEM :** Windows 10 or any other version
- **PROGRAMMING LANGUAGE** : C

CHAPTER 2

DATA STRUCTURES

A data structure could be a specialised format for organizing, processing, retrieving and storing information. While there are so many basic and advanced structure varieties available , any system is intended to rearranges information to suit a selected purpose in order that it may be accessed and worked with in appropriate ways.

In programming, a data structure may be selected or designed to store data for the purpose of working on it with various algorithms.

Each data structure contains information about the data values, relationships between the data and functions that can be applied to the data.

Characteristics of data structures

Data structures are often classified by their characteristics.

Possible characteristics are:

Linear or non-linear: This characteristic describes whether the data items are arranged in chronological sequence, such as with an array, or in an unordered sequence, such as with a graph.

Homogeneous or non-homogeneous: This characteristic describes whether all the data items in a given repository are of the same type or of various types.

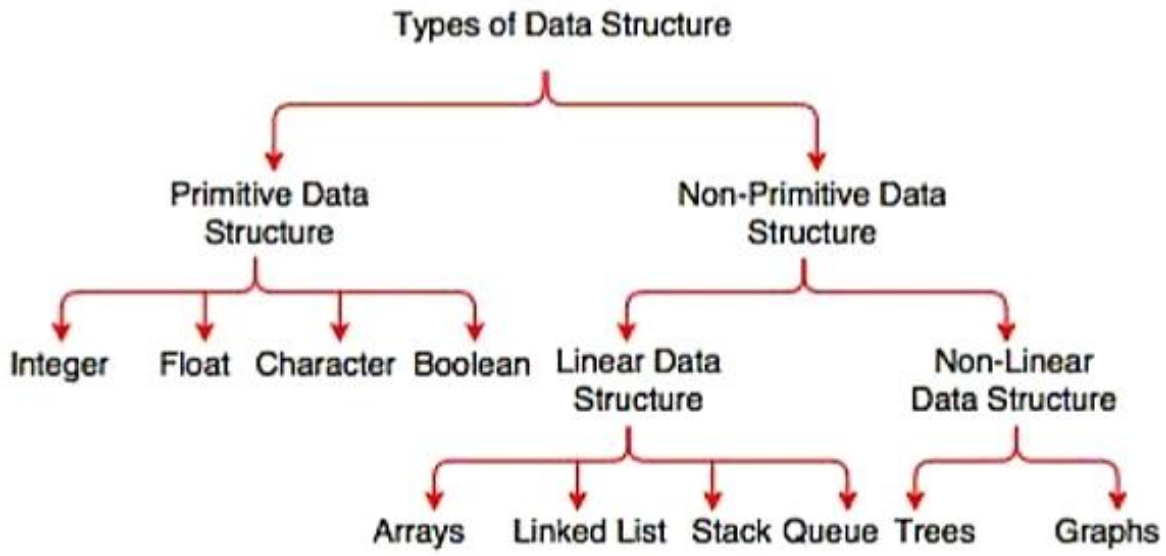
Static or dynamic: This characteristic describes how the data structures are compiled. Static data structure have fixed sizes, structures and memory locations at compile time.

Dynamic information structures have sizes, structures and memory locations that can shrink or expand depending on the use.

Depending on the circumstances and requirements of the problem one may have to use the correct data structure which will satisfies the requirements as well as the cost must be less in implementing that data structure in real time.

Types of data structures

Data structure varieties is determined by what varieties of operations is needed or of what styles of algorithms are about to be applied.



2.a Types of data structures

These types include:

Arrays- An array stores a collection of items at adjoining memory locations. Items that are the same type get stored together so that the position of each element can be calculated or retrieved easily. Arrays can be fixed or flexible in length.

Stacks- A stack stores a collection of items in the linear order where the elements are applied. This order could be last in first out (LIFO) or first in first out (FIFO).

Queues- A queue stores a collection of items similar to a stack; however, the operation order can only be first in first out.

Linked lists- A linked list stores a collection of items in a linear order. Each element, or node, in a linked list contains a data item as well as a reference, or link, to the next item in the list.

Trees- A tree stores a collection of items in an abstract, hierarchical way. Each node is linked to other nodes and can have multiple sub-values, also known as children.

Graphs- A graph stores a group of things in a very non-linear fashion.

Graphs are made up of a finite set of nodes, also known as vertices, and lines that connect them, also known as edges. These are useful for representing real-life systems such as computer networks.

Hash tables- A hash table, or a hash map, stores a collection of items in an associative array that plots keys to values. A hash table uses a hash function to convert an index into an array of buckets that contain the desired data item.

The data structure which are used in implementation of OTP and CAPTCHA GENERATION is as follows

1.linked list

2.Arrays

3 Queue

2.1 LINKED LIST

A connected list may be a sequence of information structures, that area unit connected along via links.

Linked List may be a sequence of links that contains things.

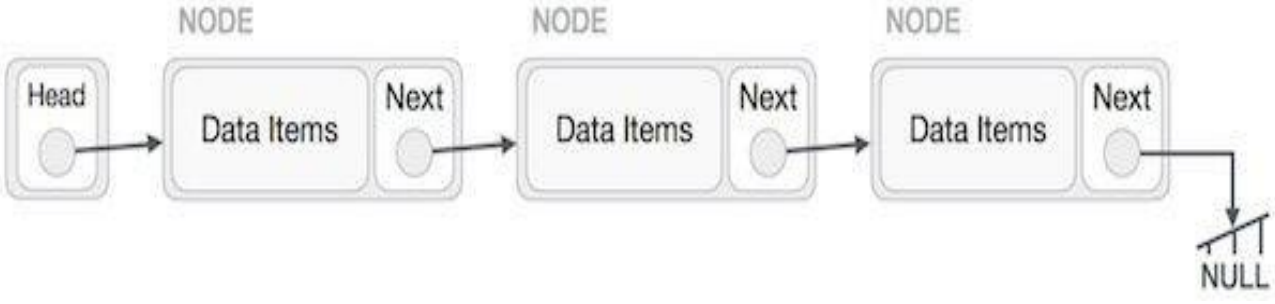
Each link contains a connection to another link.

Linked list is that the second most-used organization after array.

Following square measure the necessary terms to grasp the construct of connected List.

- Link − Each link of a linked list can store a data called an element or item or data.
- Next − Each link of a linked list contains a link to the next link called Next.
- linked List − A linked List contains the affiliation link to the primary link referred to as start.

2.b Linked List Representation



Linked list is unreal as a series of nodes, where every node points to the next node. The last node of the linked list contains Null characters which terminates the linked list. As per the mentioned illustration, following are the important points to be considered.

- Linked list is collection of nodes.
- Linked List contains a link element called head or start.
- Each link or node carries a data field(s) and a link field called next.
- Each link or node is linked with its next link using its next link.
- Last link carries a link as null to mark the top of the list.

Types of Linked List

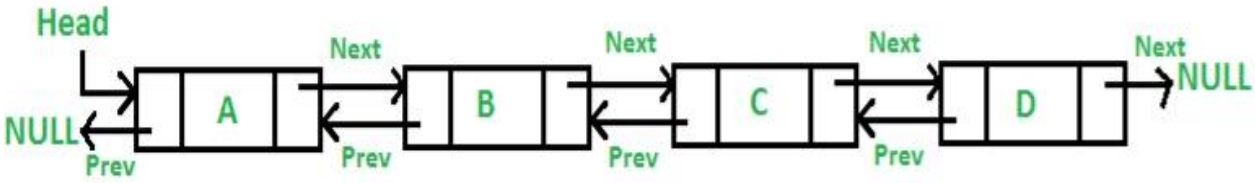
Following are the various types of linked list.

- single linked List − Item navigation is forward solely. The linked list is unidirectional and the last node points to the NULL which terminates the list. The node traversing is done in only one direction.

2.c singly linked list

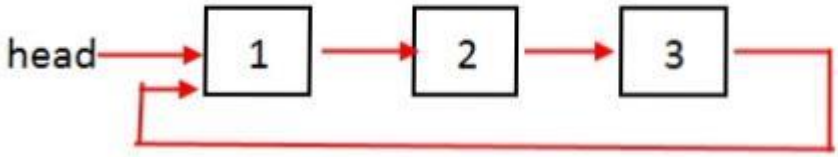
- Doubly linked List − things are often navigated forward and backward. The linked list is bidirectional and each has three parts previous address, data and next address.

2.d doubly linked list



● Circular Linked List − Last item contains, link of the first element as next part and the first element has a link to the last element as previous. The linked list forms a circular pattern.

2.e circular linked list



Basic Operations

Following area unit the essential operations supported by a listing.

● Insertion − Adds an element at the beginning of the list.

● Deletion − Deletes a component at the start of the list.

● Display − Displays the complete list.

● Search − Searches an element using the given key.

● Delete − Deletes an element using the given key.

Arrays

One memory block is allotted for the whole array to carry the weather of the array. The array parts will be accessed in constant time by the index of the actual part because the subscript.

Applications of Arrays

1. Stores Elements of Same Data Type

Array is used to store the number of elements belonging to same data type or are of similar type.

int arr[4];

Above array is used to store the four integer numbers in an array in continuous sequence.

arr[0] = 10;

arr[1] = 20;

arr[2] = 30;

arr[3] = 40;

arr[4] = 50;

Similarly if we declare the character array then it can hold only a single character. So in short if we declare character array it can store character variables while floating array stores only floating value numbers.

2. Array Used for Maintaining multiple variable names using only a single variable name and indices.

Suppose we need to store 5 roll numbers of students then without declaration of array we need to declare following – int roll1,roll2,roll3,roll4,roll5;

Now in order to get roll number of first student we need to access roll1.

Guess if we need to store roll numbers of 100 students then what will be the procedure. Maintaining all the variables and remembering all these things is very difficult and lot of variables to declare.

Consider the Array :

int roll[5];

So we are using array which can store multiple values and we have to remember just single variable name.

3. Array Can be Used for Sorting Elements

We can store the elements to be sorted in an array of a variable and then by using different sorting technique we can sort the elements in the array of elements.

Different Sorting Techniques are :

Bubble Sort

Insertion Sort

Selection Sort

Bucket Sort etc

4. Array Can Perform Matrix Operation

Matrix operations can be performed using the array as data structure. We can use 2-D array to store the matrix. Matrix operations Like adding two matrix, subtracting two matrix, matrix multiplication etc can be implemented using the arrays.

[box]Matrix can be multi-dimensional[/box]

5. Array Can be Used in CPU Scheduling

CPU Scheduling is generally managed by Queue. Queue and stacks can be managed as well as it can be implemented using the array. Array may be allocated dynamically i.e at run time.

6. Array Can be Used in Recursive Function

When the function calls another function or the same function again recursively then current values are stores onto the stack and those values will be retrieve when control comes back. This is similar to the operation like stack using arrays.

Disadvantage of Arrays:

(1) the scale of the arrays is fixed: thus we have a tendency to should apprehend the higher limit on the amount of parts before. Also, generally, the allotted memory is up to the higher limit regardless of the usage, and in practical uses, the upper limit is rarely reached.

(2) Inserting a replacement parts in associate degree array of parts is difficult as a result of new array elements space should be created for the new parts and to form room existing parts have to be shifted.

For example, suppose we maintain a sorted list of user ids in an array uid[].

uid[] = [1000, 1010, 1050, 2000, 2040, …..].

And if we would like to insert a replacement ID 1005, then to maintain the sorted order, we have to move all the elements after 1000 (excluding 1000).

Deletion is additionally high-ticket with arrays till unless some special techniques used.

For example, to delete 1010 in uid[], everything after 1010 has to be moved.

Advantage of linked list

Extensive manipulation: We can perform any variety of advanced manipulation with none previous plan of the memory house accessible (in stacks and queues we sometime get overflow conditions were we can't add elements . Here no such problem arises.)

Arbitrary Memory locations: Here the memory locations are need not be consecutive. They may be any arbitrary values. But even then the accessing of those things is simpler as every knowledge item contains at intervals itself the address to following knowledge

item. Therefore, the elements in the linked list are ordered not by their physical locations but by the logical links stored as part of the data with the node itself.

Memory can be physically released using the free function for latter allocation of nodes.

The size of the linked list is not fixed it can be altered as the memory allocation is done using the dynamic memory allocation. The linked list can grow or shrink during run time.

During run time if the user want to add or delete node it is very easy using the dynamic memory allocation and free function.

Linked lists have following drawbacks:

1) Random access is not allowed.We have to access parts consecutive ranging from the primary node. So we have a tendency to cannot do a binary search with coupled lists.
2) Additional memory house for a pointer is needed with every part of the linked list to point the next node of the linked list.

3) Arrays have higher cache neighborhood that may build a fairly huge distinction in performance.

4) Arrays are more easy to implement.

5) Reverse traversing is difficult in linked list.

Applications of linked list in real world

1. Image viewer :– Previous and next images are linked, hence can be accessed by next and previous button.
2. Previous and next page in web browser :– We can access previous and next urls are searched in web browser and others by pressing back and next button since, they are linked as linked list.
3. Music Player :– Songs in music player are linked to previous and next song.
you can play songs either from beginning or ending of the list.

4. Linked list can be used to implement stacks and queues.

5. Linked list can be used to implement graphs(Adjacency list representation of graphs).

6. Implementing hash table where each bucket of the hash table can itself be a linked list (open chain hashing).

7. A polynomial can be represented in an array or in a linked list by simply storing the co efficient and the exponent of each term.

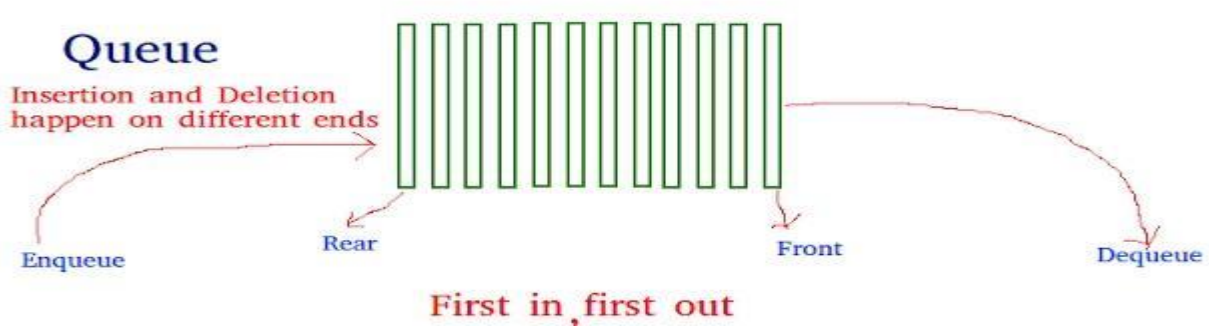
8.All the running applications are kept in a circular linked list and the operation system gives a fixed time slot to all for running in the operating system.

9. The operating system keeps on iterating over the linked list until all the applications are completed in the processing table.

2.2 QUEUES

A Queue could be a linear list of components within which deletion of a component will crop up solely at one finish referred to as the front and insertion will crop up on other end which is termed as the rear.

In the construct of a queue, the primary part to be inserted within the queue are the primary part to be deleted or faraway from the list. So, queue is said to follow FIFO(First In First Out) structure.



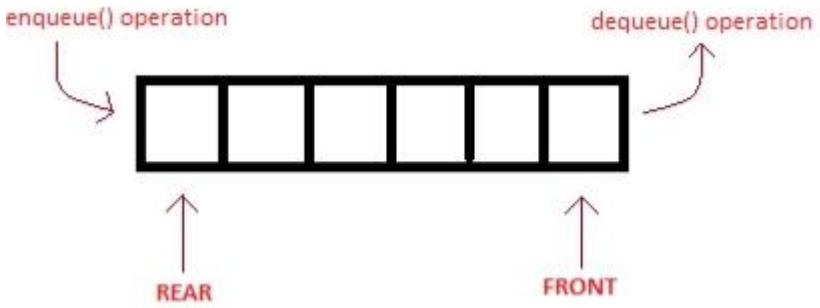
2.f QUEUE Representation

Types of QUEUES

- Linear queue
- Circular queue
- Priority queue
- Double ended queue

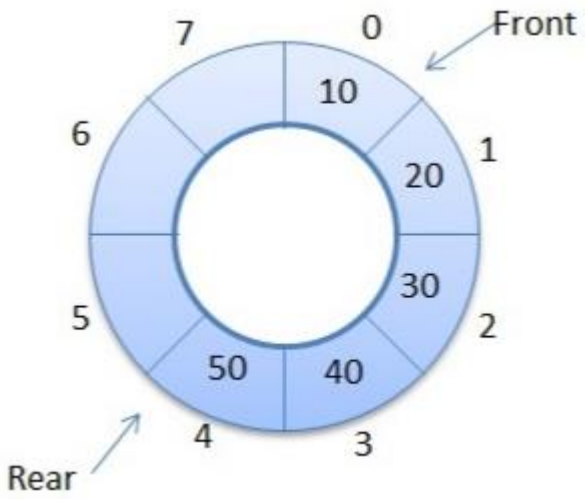
Linear queue: The insertion of elements takes place at the rear pointer and the deletion of an element takes place at the front pointer of the queue. The queue is unidirectional and the memory space available will be restricted while insertion and deletion happens for some n times as the queue reaches the max size.

2.g linear queue representation



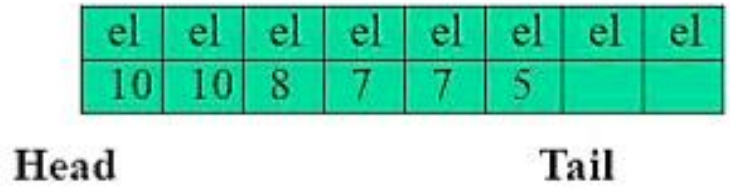
Circular queue: The insertion and deletion of elements are done in such a way that after rear pointer the front pointer comes in the queue. The memory space wastage in the linear queue is avoided in this circular queue as the front pointer follows the rear pointer in the queue.

2.h circular queue representation



Priority queue: The insertion happens at the rear pointer and the deletion is done for highest priority element in the queue. The queue where each element is associated or assigned with some priority and the element which have higher priority is served first then the next priority element.

2.i priority queue representation



Double ended queue: The queue is bidirectional and it has two its sub categories where insertion or deletion is restricted at one end of the queue.

1.Insertion restricted double ended queue

2.Deletion restricted double ended queue

2.j double ended queue representations

Basic operations performed on QUEUES

Operations might involve initializing or shaping the queue, utilizing it, and after work done completely erasing it from the memory.

Here we have a tendency to shall attempt to perceive the essential operations related to queues.

o enqueue() − add (store) an item to the queue.

o dequeue() − remove (access) an item from the queue.

Few additional functions are needed to create the above-named queue operation economical.

These are

o peek() − Gets the element at the front of the queue without removing it.

o isfull() − Checks if the queue is full.

o isempty() − Checks if the queue is empty.

Drawback of array implementation

Although, the technique of making a queue is straightforward, but there are some drawbacks of using this technique to implement a queue.

Memory wastage : The house of the array, which is used to store queue elements, can never be reused to store the weather of that queue as a result of the weather will solely or always be inserted at side and therefore the price of front can be thus high so, all the space before that, can never be filled which results memory wastage which is not desirable in working with lot of variables and functions. To overcome this one may choose the linked list Representation of the queue which will help in saving the memory space of the computer or system.

Applications of Queue

Applications of Queue, because the name suggests is employed whenever we want to manage associatey cluster of objects in an order in which the primary one returning in, also gets out first while the others await for their turns or chance, like in the following scenarios:

1. Serving requests on one single shared resource, like a printer, CPU task scheduling.
2. In real life scenario, queue are used in Call Center hone systems uses Queues to hold people calling them in an order, until a service representative is free.
3. Handling of interrupts in real - time systems. The interrupts are handled within the same order as they arrive. i.e initial return initial served.
4. When ever data is transferred asynchronously between two processes eg. IO buffers.
5. In recognition of palindromes.
6. Round robin scheduling, job scheduling and keyboard buffers.

CHAPTER 3

DESIGN

3.1 ALGORITHM

ALGORITHM TO INSERT A NODE AT THE END

Algorithm:

Step 1: Create memory block using malloc function. If malloc function are used for creating the memory then it will return address of this memory block.Otherwise memory overflow problem.

Step 2: Created memory block assign value and next pointer value or address.

Step 3: T he last node of the linked list is traversed for the attachment of this new created memory to end position. If linked list are empty then assign this address of root pointer of given linked list.

Suppose following data are inserting in linked list.

Input :201 202 203 204 205 206 207 208

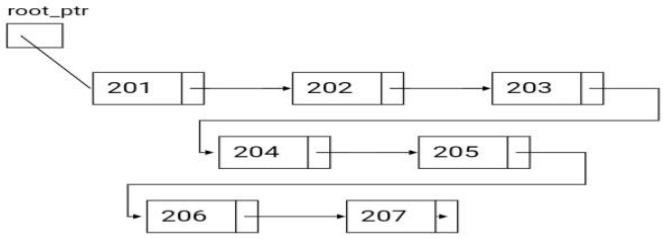
Insertion process:

Insert first element 201 of linked list. Note that initial linked list are empty. Created memory block and assign data 201 and next pointers value is NULL. Assign this first address of root .

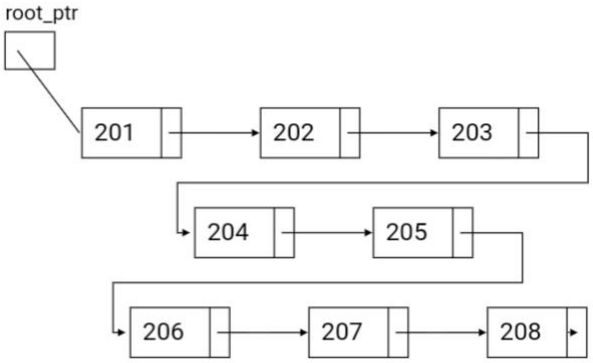
3.a linked list creation

Similar way to other node 203,204,205,206, 207. After insert linked list is

3.b insertion element by element



Insert last node 208.



3.c last insertion of element

ALGORITHM TO GENERATE OTP

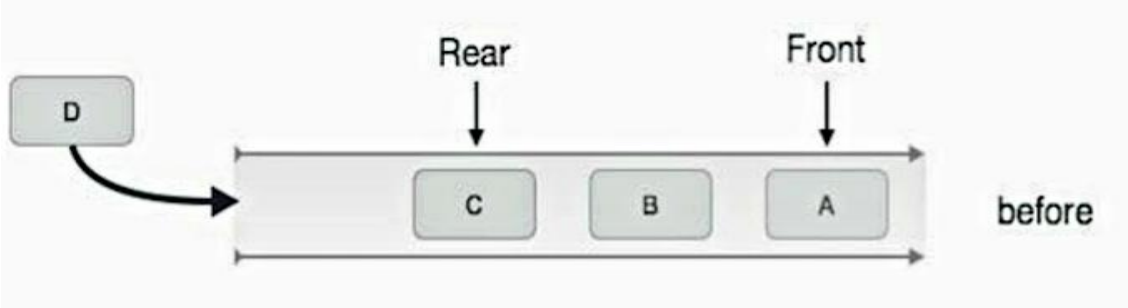
- Begin
- User must enter the number of digits.
- The memory is allocated for the nodes using dynamic allocation **malloc( )** function.
- The data part of the linked list is generated using **random( )**.
- The random number generated is given as data for each node.
- A loop is used for n times inputted by the user.
- The link part of the linked list is populated by the new nodes address.
- This process repeats until n times to produce n digit pin.
- The data part in the linked list is displayed one after the other.
- End

ALGORITHM FOR INSERTION OF DATA INTO QUEUE

Two pointers – **front** and **rear** are maintained by Queues and hence the operations are different from that of Stacks.

The steps to enqueue (insert) value or data into a queue are -

- **Step1** − Check if the queue is full.
- **Step 2** − If the queue is full, produce overflow error and exit.
- **Step3** − If the queue is not full, increment rear pointer to point the next space.
- **Step4**− Add data element to the queue location, where the rear is pointing.



3.d queue before insertion

3.e queue after insertion

ALGORITHM FOR GENERATION OF CAPTCHA

- Begin
- Get the of number of characters required by the user.
- A **for loop** is used for n times inputted by the user.
- A random number is created using **random()** function used for switching case.
- The rear pointer of the queue is incremented .
- **Switch case** is used for different cases like digits, lowercase alphabet, uppercase alphabet.
- The random digit is generated by adding a random number to **ASCII** value of 0 that is 48.
- The random lowercase alphabet is generated by adding a random number to **ASCII** value of lowercase **'a'** that is 97.
- The random uppercase alphabet is generated by adding a random number to **ASCII** value of uppercase **'A'** that is 65.
- The generated characters is inputted in the queue.
- End

3.2 PSEUDO CODE

PSEUDO CODE FOR INSERTION OF NODE AT END OF THE LINKED LIST

//The condition (head == NULL) gets satisfied. Hence, we just need to allocate the space for the node by using malloc statement in C and C++. Data and the link part of the node are modified or changed by using the following steps or statements.

ptr->data = item;

ptr -> next = NULL;

//Initially if only one node is present .

Since, ptr is that the solely node that may be inserted within the list therefore, we need to make this node pointed by the head pointer of the list. This whole work will be done by using the following statements.

Head = ptr

//The condition Head = NULL would fail as it points to first node of the linked list, since

If Head is not NULL, the linked list already created. Now, declare a temporary pointer temp in order to traverse through the linked list, temp is made to point the first node of the list and also for the traversing of nodes.

Temp = head

//Then, traverse through the whole linked list victimization by the statements:

while (temp→ next != NULL)

temp = temp → next;

//At the tip of the loop, the temp will be pointing to the last node of the created list. Now, allocate the memory for the new node, and assign the value or data to its data part.

Since, the new node goes to be the last node of the list hence, successive a part of this node has to be inform to the null. We need to make the next part of the temporary worker node (which is presently the last node of the list) purpose to the new node.

temp->next=ptr;

ptr->next = NULL;

PSEUDO CODE FOR INSERTION OF ELEMENTS INTO QUEUE

```
 if(rearpt == MAXSIZE-1)
 //Checking whether the queue having space to insert new element.
printf("\nQueue is Full!!");
 //If queue is full an error message is printed.
 //If queue has space for insertion else part is carried out.
  else
 {
          //If the front pointer is -1(for first element insertion) it is made as 0.
          if(frontpt == -1)
           frontpt= 0;
            //The rear pointer of the queue is incremented by 1 and the element is inserted
           into  the array of queue.
            rearpt++;
            items[rearpt]=data;
  }
```

CHAPTER 4

IMPLEMENTATION

4.1 RANDOM( ) FUNCTION

Pseudo Random Number Generator(PRNG) refers to associate degree algorithmic rule that uses mathematical formulas to supply sequences of random numbers.PRNGs generate a sequence of numbers which are approximating the properties of randomness in the numbers. A PRNG starts from associate degree absolute beginning state employing a seed state. Many numbers area unit generated in a very short time and might even be reproduced later, if the starting point in the sequence is known.

Characteristics of PRNG

Efficient : PRNG will turn out several numbers in a very short time and is advantageous for applications that require several numbers.

Deterministic : A given sequence of number can be reproduced at a later date if the starting point in the sequence is known as determinism is hardly if you need to replay the same sequence of number again at a later stage.

Periodic : PRNGs are periodic in nature, which means that the sequence will eventually repeat itself after certain time period. While periodicity is hardly even a desirable character modern PRNGs have a period that is so long that it can be ignored for most practical purposes.

A **linear congruential generator** (LCG) is associate degree algorithmic rule are used to yields a sequence of pseudo-randomized numbers calculated with a discontinuous piecewise of a linear equation. The method represents one of the oldest and best-known pseudorandom number generator algorithms The theory behind them is relatively easy to understand, and they are easily to be implemented and fast, especially on computer hardware which can provide or requires modulo arithmetic by storage-bit truncation.

The generator is defined by recurrence relation:

$X_{n+1} = (aX_n + c) \bmod m$

Where X is the sequence of pseudorandom values, and

m, 0 < m — the "modulus"

a, zero < a < m — the "multiplier"

c, 0 < c < m — the "increment"

X 0, zero < X zero < m — the "seed" or "start value"

Applications of PRNG

PRNGs area unit appropriate for applications wherever several random numbers area unit needed and wherever it's helpful that the same sequence can be replayed easily. Popular examples of such applications are simulation and modelling applications.PRNGs aren't appropriate for the applications wherever it's necessary that the numbers of area unit really unpredictable, such as data encryption and gambling.

4.2 PROGRAM FOR GENERATING OTP

```
 struct node
{
        int d;
       struct node * nt;
};
struct node *start=NULL,*cur;
main( )
{
        int i, data;
       printf("enter the no of digits or characters:");
       scanf("%d",&n);
      head=(struct node *)malloc(sizeof(struct node));
      cur=start;
```

```
 for(i =0 ;i < n; i++)
{
      if(i < n-1)
     {
            data= random(9);
           //data part of linked list is populated by the random
           cur->nt=(struct node *)malloc(sizeof(struct node));
           cur=cur->nt;
     }
 //for last insertion of elements or data into the linked list.
 else if(i==n-1)
     {
           data=random(9);
           cur->d=data;
           cur->nt=NULL;
      }
 }
//printing of data part of linked list one after the other.
 cur=start;
while(cur!=NULL)
{
     printf("%d",cur->d);
     cur=cur->nt;
}
```

TRACING OF THE CODE

- The number of digits or characters is inputted from the user.
- The head node is created by the **malloc** function.
- The current pointer is initialized with the **head**.
- A **for** loop is used for creating linked list.
- Memory allocation for new node is done using dynamic memory allocation.
- The linked list is created for the given number of digits.
- The data part is generated by the **random( )** function**.**
- The random number varies from 0 to 9.
- The address part is populated by address of next node.
- These process continues until **for** loop fails.
- The **for** loop runs for the inputted number of times.
- After the completion of **for** loop the linked list is ready.
- The linked list of random numbers is generated.
- The data part of the linked list contains the required **OTP.**
- The data part of the linked list is displayed to the user.
- The nodes in the linked list is traversed one after the other using **while** loop.

4.3 PROGRAM FOR GENERATING CAPTCHA

```
main( )
 {
       //get the number of characters from user.
        while(r!=x)
        {
              r++;
             ch=random(3);
             switch(ch)
            {
                    case 1:
```

```
                    a[r]=65+random(26);
                break;
            case 2:
                    a[r]=48+random(10);
                    break;
            case 3:
                a[r]=97+random(26);
                    break;
            default:
                    a[r]=97+random(26);
                break;
        }
    for(i = f ;i < r ; i++ )
      printf("%d", a[i]);
     //print the elements of queue to user
}
```

TRACING OF THE CODE

- The number of characters is inputted by the user.
- A **while** loop is used for generating the **CAPTCHA** .
- The rear pointe is incremented by one.
- The **random** function is used to generate a number from 0 to 1 which is used for switching.
- Switching is done by using **switch case** statement**.**
- Switch case has 3 cases.
- First case to generate a uppercase alphabet varying from A to Z.
- Second case to generate a digit varying from 0 to 9.
- Third case to generate a lowercase alphabet varying from a to z.
- Default case is also to generate lowercase alphabet.

- The generated characters is inputted in queue at rear pointer.
- This process continues until the **while** loop fails.
- The **while** loop fails once the required number of characters is inputted into queue.
- The required **CAPTCHA** is ready.
- The elements in the queue from front pointer to rear pointer is displayed using a **for** loop.
- The final queue will contain alphanumeric characters varying from 0 to 9 , a to z and A to Z.

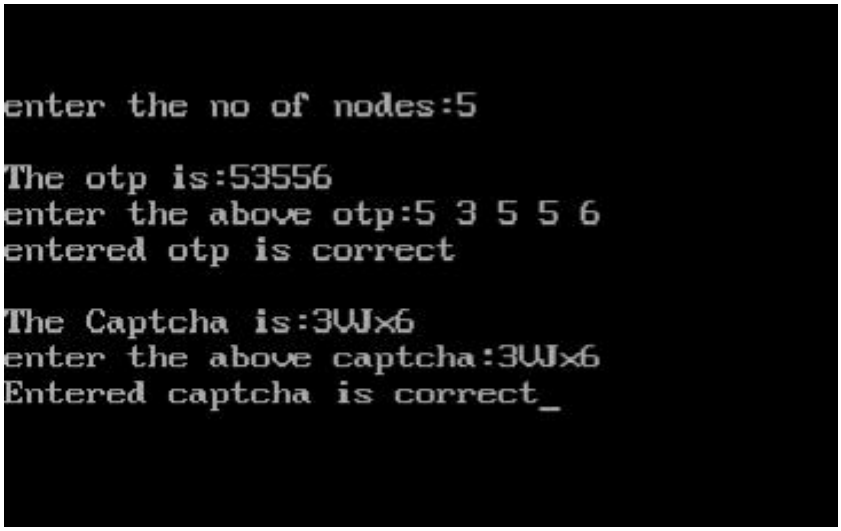
CHAPTER 5

RESULTS

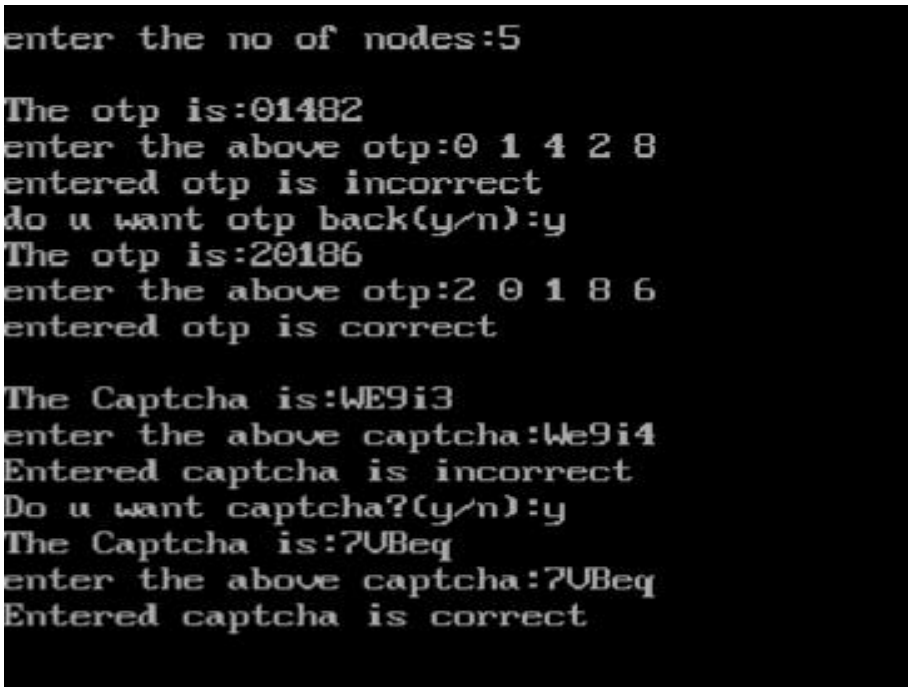
- The user must input the required number of digits or characters.
- The program executes and produces the output.
- The output contains both **OTP** and **CAPTCHA** of required digits or characters.
- The **OTP** is generated using linked list.
- The **CAPTCHA** is generated using linear queue.
- The current output is different from the previous outputs.
- Each time run the program we get different **OTP** and different **CAPTCHA .**
- Even for same number of required digits or characters the output is different in different running of the program.

SAMPLE OUTPUTS

The user must input the number of digits or characters first. The generated OTP will be displayed and user have to input the OTP correctly in the verification part. If the user inputs the OTP correctly then a message is printed that OTP entered is correct and then it continues to the CAPTCHA generation. The generated CAPTCHA is displayed and the user must input the CAPTCHA correctly in the verification part. If the user inputs correctly then a message is printed that entered CAPTCHA is correct and the program finishes.



The user must input the number of digits or characters first. The generated OTP will be displayed and the user must input the OTP correctly for verification. If the user inputs the OTP incorrectly an error message is printed that the entered OTP is incorrect. A statement is printed whether the user want's a new OTP. If the user wishes to get new OTP then again a new OTP is generated and displayed for verification. If the user enters the correct OTP then it continues to the CAPTCHA generation. The generated CAPTCHA is displayed and user is asked to enter the CAPTCHA correctly. If the user inputs incorrectly then an error message is printed that the entered CAPTCHA is incorrect and a statement printed whether the user want's to continue if so new CAPTCHA is generated and displayed. If the user enters correct CAPTCHA then the program finishes.



CHAPTER 6

CONCLUSION

- The main objective of the problem statement is achieved.
- All the objectives and expected outcomes are full filled.
- **OTP** and **CAPTCHA** is generated for required number of digits or characters.
- Generation of **OTP** is implemented using single linked list.
- The generated **OTP** contain digits varying from 0 to 9.
- The output has random n digits.
- Generation of **CAPTCHA** is implemented using linear queue.
- The generated **CAPTCHA** contains alphanumeric characters varying from 0 to 9,a to z and A to Z.
- The output has combinations of all 3 possible characters like 0 to 9,a to z and A to Z.
- The current output generated is different from previous outputs.
- The program generates different output each time u run the program.
- The output is different even the inputted number of digits or characters is same.
- The dynamic nature of program is achieved.
- The randomness of output is achieved using PNRG(Pseudo Number Random Generator) algorithm function **random(max)** function.
- Even though the cycle repeats in the PNRGs the time period is so long, so that it can be ignored.

REFERENCE

1.”Data structures with C",SEYMOUR LIPSCHUTZ, special indian edition, thirteenth reprint 2015, McGraw Hill Education.

2. “Data structure using C “, Aaron M. Tanenbaum, Yedidyahlangsam and Moshe J Augenstein Thirteenth impression 2014, Pearson Education.

3. “Data structure - A Pseudo code approach with C”, Richard F Gilberg and Behrouc A Forouzan, second edition, fifth Indian reprint 2015, Cengage learning.

4. Online websites related data structure.