**DATA STRUCTURE MINI PROJECT**

**LOTTERY SYSTEM**

**TEAM MEMBERS:**

* Mr. Pratik Panchal (60004188006)
* Mr. Prem Bhajaj (60004188001)
* Ms. Rucha Shinde (60004188022)

**DATA STRUCTURES:**

1. Simple Linked list
2. Queue using Linked list.

**ABSTRACT:**

A lottery is a form of gambling which involves the drawing of lots for a prize. Lottery is outlawed by some governments, while others endorse it to the extent of organizing a national or state lottery. In this a random token is generated, if it matches with the one you have purchased, and then you are the Winner. The generation of such tokens are purely random which makes it interesting and exciting for the user. Though little luck factor can help you. Here, we are trying to imitate lottery system at a smaller level.

The operations performed in this comprise of:

1. Generation of tickets,

2. Selling the tickets and

3. Generating the winning token

4. Displaying available and sold tickets

1. Generation of tickets

The ticket values have a static range of 15 digits, which is generated in the following way:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 18 | 10 | 22 | 23 | 20 | 00000 |
| yy | mm | dd | Hr | MM | serial |

The first two digits of it will be year, the next 2 i.e. 10 is month, the next to it is date, and so on, the last 5 digits are generated serially.

2. Selling the tickets

The admin who is managing the lottery system can check the available tickets. The tickets generated were present in the linked list. The ones which are sold are removed from the list and added to a queue.

3. Generating the winning token

The winning token is selected from the tickets which are sold. The sold out tickets are present in the queue, a random index number of it selected of the queue the ticket residing in it will be our winning token.

4. Displaying available and sold tickets

The admin can check the tickets which are sold and also he can check the tickets which are available.

**SOURCE CODE:**

#include<stdio.h>

#include<conio.h>

#include<time.h>

long count;

int choice;

long soldTicketCounter;

struct timespec {

time\_t tv\_sec; /\* seconds \*/

long tv\_nsec; /\* nanoseconds \*/

}ts;

struct node{

long long int TicketNo;

struct node \*next;

}\*start, \*newnode, \*temp, \*pre, \*soldF, \*soldR;

////Comment this if using TurboC

void clrscr(){

system("cls");

}

////

long randnum(long bound){

long r = 0;

int i;

for(i = 0; i < 10; i++){

clock\_gettime(1, &ts);

srand((time\_t)ts.tv\_nsec);

r += rand();

}

return r % bound;

}

void insertIntoSellQueue(struct node \*n){

if(soldF == NULL || soldR == NULL){

soldF = n;

soldR = n;

n->next = soldF;

}

else{

n->next = soldF;

soldR->next = n;

soldR = soldR->next;

}

soldTicketCounter++;

}

void insertNewTicket(long long int TicketNo){

newnode = malloc(sizeof(struct node));

newnode->TicketNo = TicketNo;

newnode->next = NULL;

if(start == NULL){

start = newnode;

temp = start;

}

else{

temp->next = newnode;

temp = temp->next;

}

}

void trashTickets(){

while(start != NULL){

temp = start;

start = start->next;

temp->next = NULL;

free(temp);

}

if(soldR != NULL){

soldR->next = NULL;

}

while(soldF != NULL){

temp = soldF;

soldF = soldF->next;

temp->next = NULL;

free(temp);

}

}

void sellTicket(long loc){

if(start != NULL){

if(loc > 1){

count = 1;

temp = start;

while(count < loc){

count++;

pre = temp;

temp = temp->next;

}

pre->next = temp->next;

}

else{

temp = start;

start = start->next;

}

insertIntoSellQueue(temp);

}

else{

printf("Tickets not generated yet");

}

}

void showAvailableTickets(){

printf("-------------------\n");

if(start != NULL){

temp = start;

count = 0;

printf("Sr no\tTicket Number\n");

while(temp != NULL){

printf("%li\t: %lld\n", ++count, temp->TicketNo);

temp = temp->next;

}

}

else{

printf("Tickets Not Generated\n");

}

printf("-------------------\n");

}

void showSoldTickets(){

printf("-------------------\n");

if(soldF != NULL){

temp = soldF;

count = 0;

printf("Sr no\tTicket Number\n");

do{

printf("%li\t: %lld\n", ++count, temp->TicketNo);

temp = temp->next;

}while(temp != soldF);

}

else{

printf("No tickets Sold\n");

}

printf("-------------------\n");

}

void generateLotteryTickets(){

int LotteryTickets, i;

long long int num, initial;

time\_t now;

int hours, minutes, day, month, year;

trashTickets();

soldTicketCounter = 0;

printf("Enter number of lottery tickets : ");

scanf("%d",&LotteryTickets);

time(&now);

struct tm \*local = localtime(&now);

hours = local->tm\_hour;

minutes = local->tm\_min;

day = local->tm\_mday;

month = local->tm\_mon + 1;

year = (local->tm\_year + 1900) % 100;

num = year;

num \*= 100;

num += month;

num \*= 100;

num += day;

num \*= 100;

num += hours;

num \*= 100;

num += minutes;

num \*= 100000;

initial = num;

for(i = 0; i < LotteryTickets; i++){

insertNewTicket(num++);

}

printf("Lottery tickets generated from %lld to %lld.",initial, num);

}

void displayAndSell(){

long loc;

showAvailableTickets();

printf("Enter serial number to be selled: ");

scanf("%li",&loc);

sellTicket(loc);

}

long long int getWinner(){

long luckyNumber = randnum(soldTicketCounter);

count = -1;

temp = soldF;

while(count < luckyNumber){

temp = temp->next;

count++;

}

return temp->TicketNo;

}

int main(){

start = newnode = temp = pre = soldF = soldR = NULL;

while(1){

clrscr();

printf("Enter choice:\n");

printf("1. Generate Lottery Tickets\n");

printf("2. Display Aailable Tickets\n");

printf("3. Display Sold Tickets\n");

printf("4. Sell Ticket\n");

printf("5. Get Winning Lottery Number\n");

printf("6. Exit\n");

choice = getch() - 48;

clrscr();

switch(choice){

case 1:

generateLotteryTickets();

break;

case 2:

showAvailableTickets();

break;

case 3:

showSoldTickets();

break;

case 4:

displayAndSell();

break;

case 5:

printf("The Lucky Winner is >>>%lld<<<",getWinner());

break;

case 6:

exit(0);

break;

default:

printf("Wrong Option Selected\n");

break;

}

getch();

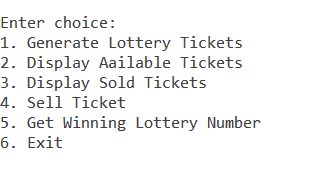
}

return 0;

}

**OUTPUT:**

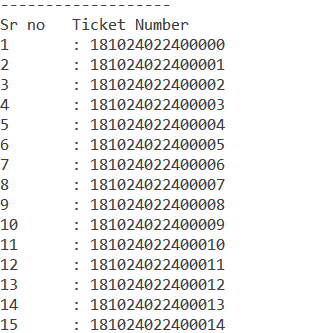
**Main Page**



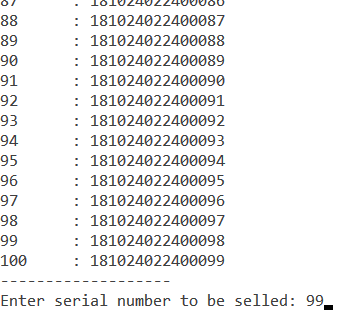
**Generating Lottery Tickets**



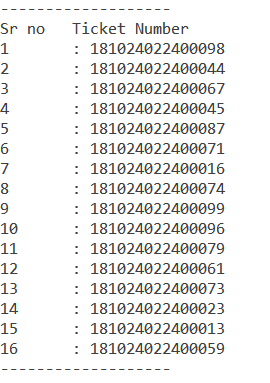
**Display Available Tickets**



**Selling of Tickets**



**Displaying Sold Tickets**



**Randomizing the sold out tickets**

