```
/*
        ***BANK ATM SYSTEM***

    GROUP NAME : DEBUG DEMUNS

    NAME                UIN             ROLL NO
  1) MUZAMMIL ANSARI    241A035         29
  2) TANISHQ BORASTE    241A031         26
  3) HARSHIT MISHRA     241A050         41
  4) NITESH MAHTO       241A044         36

*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>

/*
  node structure for implementing a queue to store transaction history
*/
typedef struct node {
    char statement[50];
    struct node* link;
} node;

/*
  ATM function prototypes
*/
void pinGeneration(void);
int checkPin(void);
void showBalance(int *);
void depositMoney(node **, int *);
void withdrawMoney(node **, int *);
void saveHistory(node **, char *);
void removeHistory(node **);
void showHistory(node **);

int main(void) {
    int choice1, choice2;
    int pinValid = 0, balance = 0;

    node *head = NULL;

    while (1) {
        printf("\n\n\t***ATM System***\n=====================\n");
        printf("1. Generate PIN\n2. Use ATM\n3. Exit\n");
        printf("\nYour choice: ");
        scanf("%d", &choice1);

        switch (choice1) {
            case 1: pinGeneration();
                    exit(EXIT_SUCCESS);

            case 2: pinValid = checkPin();

                    if (pinValid) {
                        printf("\nValid PIN\n");
                    } else {
                        printf("\nInvalid PIN. Please generate a PIN if you don't
```

```
have one.\n");
                        exit(EXIT_FAILURE);
                    }

                    /*
                                On valid PIN entry by user, the ATM Menu is
presented to the user
                             */
                    while(pinValid) {
                        printf("\nATM System Menu\n===============\n\n");
                        printf("1. Check Balance\n2. Deposit\n3. Withdraw\n4. View
transaction history\n5. Quit\n\n");

                        printf("Enter choice: ");
                        scanf("%d", &choice2);

                        switch(choice2) {
                            case 1: showBalance(&balance);
                                    break;
                            case 2: depositMoney(&head, &balance);
                                    break;
                            case 3: withdrawMoney(&head, &balance);
                                    break;
                            case 4: showHistory(&head);
                                    break;
                            case 5: printf("\nThank you for using the ATM\n");
                                    exit(EXIT_SUCCESS);
                            default: printf("\nInvalid option entered!\n");
                                     break;
                        }
                    }

            case 3: exit(EXIT_SUCCESS);

            default: printf("\nInvalid choice...Try again...\n");
                     break;
        }
    }
    return 0;
}

/*
  This function will search for the PIN entered by the user
  in the file where the PIN numbers are stored
  If PIN is found return 1
  Otherwise, return 0
*/
int checkPin(void) {
    FILE *fp;
    // buffer to read PIN and store from file
    char pin[8];
    // buffer to read PIN and store from user
    char keyPin[8];
    int pinValid = 0;

    printf("\n\nEnter the PIN: \n");
    scanf("%s", keyPin);

    fp = fopen("pin.txt", "r");
```

```
    if (NULL == fp) {
        printf("\nFile cannot be opened\n");
        exit(EXIT_FAILURE);
    }

    /*
      Search for the PIN entered by user in file pin.txt
    */
    while (fgets(pin, sizeof(pin), fp) != NULL) {
        if (strstr(pin, keyPin)) {
            pinValid = 1;
        }
    }
    fclose(fp);

    return pinValid;
}

/*
  This function will generate a 4-digit random number that
  is considered as PIN
*/
void pinGeneration(void) {
    FILE *fp;

    /*
        Generate a random 4 digit number
      */
    srand(time(NULL));
    int generatedPin = 1000+rand()%9000;

    printf("\nPIN generated successfully\n");
    printf("\nYour generated PIN: %d\n", generatedPin);
    printf("\nRe-run the program and use ATM with this PIN\n\n");

    fp = fopen("pin.txt", "a");
    if (NULL == fp) {
        printf("\nCannot open file!");
        exit(EXIT_FAILURE);
    }

    /*
      Write PIN to the file
    */
    fprintf(fp, "%d\n", generatedPin);
    fclose(fp);
}

/*
  This function will display the current balance amount
*/
void showBalance(int *balance) {
    printf("\nYour current balance is Rs.%d\n", *balance);
}

/*
  This function will add the money deposited to the balance
*/
void depositMoney(node **head, int *balance) {
```

```
    int depositAmount;
    /*
      buffer to store
    */
    char depositStmt[50];

    printf("\nEnter amount to deposit: ");
    scanf("%d", &depositAmount);

    if (depositAmount > 0) {
        *balance += depositAmount;
        printf("\nRs.%d deposited\n", depositAmount);

            /*
              saving formatted string in depositStmt character array
            */
        snprintf(depositStmt, sizeof(depositStmt), "Rs.%d deposited\n",
depositAmount);
        saveHistory(head, depositStmt);
    } else {
        printf("\nInvalid amount entered\n.");
    }
}

/*
  This function will deduct the money withdrawn from the balance
*/
void withdrawMoney(node **head, int *balance) {
    int withdrawAmount;
    char withdrawStmt[50];

    printf("\nEnter amount to withdraw: ");
    scanf("%d", &withdrawAmount);

    if (withdrawAmount > 0) {
        if (withdrawAmount > *balance) {
            printf("\nCannot withdraw. Balance Rs.%d\n", *balance);
        } else {
            *balance -= withdrawAmount;
            printf("\nRs.%d withdrawn\n", withdrawAmount);

            /*
                   saving formatted string in withdrawStmt character array
                 */
            snprintf(withdrawStmt, sizeof(withdrawStmt), "Rs.%d withdrawn\n",
withdrawAmount);
            saveHistory(head, withdrawStmt);
        }
    } else {
        printf("\nInvalid amount entered\n.");
    }
}

/*
  This function will save a transaction statement
*/
void saveHistory(node **head, char *str) {
    static int count = 0;
    node *temp;
```

```
    temp = (node *)malloc(sizeof(node));

    strcpy(temp->statement, str);
    temp->link = NULL;

    if (NULL == *head) {
        *head = temp;
        count++;
    } else {
        if (10 == count) {
            removeHistory(head);
            count--;
        }
        node *p;
        p = *head;
        while (NULL != p->link) {
            p = p->link;
        }
        p->link = temp;
        count++;
    }
}

/*
  This function is used to remove the oldest transaction when
  10 transactions are made
*/
void removeHistory(node **head) {
    node *temp;
    temp = *head;
    *head = (*head)->link;
    temp->link = NULL;
    free(temp);
}

/*
  This function will display the transaction history
*/
void showHistory(node **head) {
    node *temp;
    temp = *head;

    if (NULL == temp) {
        printf("\nNo transaction history...\n");
    } else {
        printf("\nTransaction History\n-------------------\n\n");
        while (NULL != temp) {
            printf("%s\n", temp->statement);
            temp = temp->link;
        }
    }
}
```