# Project 2024 [data structure]

## Banking System:

**Create a simple banking system that uses dynamic data structures to manage customer accounts. Implement functionalities like creating accounts, deleting accounts, depositing and withdrawing funds, and checking account balances.**

**The main function should contain a loop that lets the user choose the function he needs to perform from a list of functions. For example, in the 1st project you may have the following menu.**

**1- Create account**

**2- Delete account**

**3- Deposit**

**4 -Withdraw**

**5- Check account balance**
**0 – Exit**

## Header File:-

```
#include <string>
using namespace std;
struct Account
{
	int id;
	string name;
	double balance;
	Account *next;
};
class Bank
{
private :
	Account * head;
	int i;
public :
	// no-argument constructor
	Bank();
	//add account
	void add_acc ();
	void delete_acc(int id);
	bool search (int id) const;
	// show balance
	void display_acc(int id) const;
	//
	void display_all () const;
	// deposit
	void deposite(double amount , int id);
	//withdraw
	void withdraw (double amount , int id);
	// isEmpty
	bool Is_empty () const;
	// is_full
	bool Is_Full () const;
};
```

**Main :**

```
#include<iostream>
#include<string>
#include<cmath>
#include<iomanip>
#include "bank_header.h"
using namespace std;
Bank::Bank()
{
	head = NULL;
	i = 0;
}
void Bank::add_acc()
{
	if (this->Is_Full())
	{
		cout << "Can't add new account" << endl;
	}
	else
	{
		Account * new_acc = new Account;
		cout << "Enter name :" << endl;
		cin >> new_acc->name;
		new_acc->id = ++i;
		new_acc->balance = 0;
		new_acc ->next = NULL;
		if (this->head == NULL)
		{
			head = new_acc;
		}
		else
		{
			Account *tmp = this->head;
			while (tmp ->next != NULL)
			{
				tmp = tmp ->next;
			}
			tmp->next = new_acc;
		}
	}
}
bool Bank::Is_empty () const
{
	return this->head == NULL;
}
bool Bank::Is_Full () const
{
	Bank * newacc;
	try
	{
		newacc = new Bank;
		delete newacc;
		return false;
	}
	catch (bad_alloc e)
	{
```

```
            return true;
        }
}
void Bank::display_all() const
{
        Account * tmp = this->head;
        while (tmp != NULL)
        {
        cout << tmp ->id << "  " << tmp ->name << "  " << tmp -
        >balance << endl;
                tmp = tmp ->next;
        }
}
bool Bank::search (int id) const
{
        Account *tmp = this->head;
        while (tmp != NULL)
        {
                if (id == tmp->id)
                {
                        return true;
                }
                tmp = tmp ->next;
        }
        return false;
}
void Bank::display_acc(int id) const
{
        if (this->search(id) == true)
        {
                Account *tmp = this->head;
                while (tmp != NULL)
                {
                        if (id == tmp->id)
                        {
        cout << tmp->id << "  " << tmp->name << "  " << tmp ->balance
        << endl;
                                break;
                        }
                        tmp = tmp ->next ;
                }
        }
        else
        {
                cout << id <<" not found" << endl;
        }
}
void Bank::deposite(double amount , int id)
{
        if (amount > 0 )
        {
                if (this->search(id) == true)
                {
                        Account * tmp = this->head ;
                        while (tmp != NULL)
                        {
                                if (tmp ->id == id)
                                {
                                        tmp->balance += amount;
        cout << "account number : "<<id << " has abalance now "<<
        tmp->balance<<endl;
                                        break;
                                }
                                tmp = tmp ->next;
                        }
                }
                else
                {
                        cout << "Invalid account number "<< id << endl;
                }
        }
        else
        {
                cout << "Invalid amount" << endl;
        }
}
```

```
void Bank::withdraw (double amount , int id)
{
	if (this->search(id))
	{
		if (amount > 0)
		{
			Account * tmp = this->head ;
			while (tmp != NULL)
			{
				if (tmp ->id == id)
				{
					if (tmp ->balance >= amount)
					{
					tmp->balance -= amount;
		cout << "account number : "<<id << " has abalance now
		"<< tmp->balance<<endl;
					}
					else
					{
						cout << "avaliable amount " << tmp -
>balance << endl;
					}
					break;
				}
				tmp = tmp ->next;
			}
		}
		else
		{
			cout << "Invalid amount " << amount << endl;
		}
	}
	else
	{
		cout << id << " not found" << endl;
	}
}
void Bank::delete_acc(int id)
```

```
{
	if (this ->search (id))
	{
		Account * tmp = head,*t;
		while (tmp != NULL)
		{
			if (tmp ->id == id)
				break;
			t = tmp;
			tmp = tmp ->next;
		}
		if (tmp  == head)
		{
			head = head ->next;
			delete tmp;
		}
		else
		{
			if (tmp != NULL)
			{
				t ->next = tmp ->next;
				delete tmp;
			}
			else
			{
				cout << id <<" Not Found" << endl;
			}
		}
	}
	else
	{
		cout << id << " not found" << endl;
	}
}
int main()
{
	int choice,id;
	double amt;
```

```
        Bank *b = new Bank;
        while (true)
        {
                cout << "Enter your choice : " <<endl;
                cin >> choice;
                switch (choice)
                {
                case 0:
                        cout << "All Accounts : "<< endl;
                        b->display_all();
                        system("pause");
                    exit(0);
                        break;
                case 1:
                        b->add_acc ();
                        break;
                case 2:
                        cout << "Enter id of Account to delete : " << endl;
                        cin >> id;
                        b->delete_acc(id);
                        break;
                case 3:
cout << "Enter id for account and amount to deposit" << endl;
                        cin >> id >> amt;
                        b->deposite(amt,id);
                        break;
                case 4:
cout << "Enter id for account and amount for withdraw" << endl;
                        cin >> id >> amt;
                        b->withdraw(amt,id);
                        break;
                case 5:
                        int id ;
                        cout << "Enter account number : " << endl;
                        cin >> id;
                        b->display_acc(id);
                        break;
                case 6:
                    b->display_all();
                    break;
                default :
                    cout << "try again " << endl;
                    break;
                }
        }
        system("pause");
        return 0;
}
```

**Library Management System:**
**Create a library management system that maintains a collection of books using dynamic data structures. Implement functionalities like adding books, searching for books by title or author, and managing borrow/return operations.**

**Header :**

```
#include <iostream>
using namespace std;
struct Book
{
        int id;
        string author,title;
        bool isborrow;
        int number_pages;
        Book *next;
};
class Library
{
private :
        Book *head;
        int i ;
public :
        Library();
        bool Is_Full () const;
        bool Is_Empty () const;
        void add_book ();
```

```
	void delete_book (int);
	bool search_id(int) const;
	void display_book(int) const;
	void display_all() const;
	void search_title(string)const;
	void search_author(string)const;
	void borrow_book(int) const;
	void return_book(int) const;
};
```

**Main**

```
#include<iostream>
#include<string>
#include<cmath>
#include<iomanip>
#include "Library.h"
using namespace std;
Library::Library()
{
	head = NULL;
	i = 0;
}
void Library::add_book()
{
	if (this->Is_Full())
	{
		cout << "Can't add new book" << endl;
	}
	else
	{
		Book * new_book = new Book;
		cout << "Enter Title : " << endl;
		cin >> new_book ->title;
		cout << "Enter Author : " << endl;
		cin >> new_book ->author;
		cout << "number of pages : " << endl;
		cin >> new_book->number_pages;
```

```
        new_book ->isborrow = false;
        new_book->id = ++i;
        new_book ->next = NULL;
        if (this->head == NULL)
        {
            head = new_book;
        }
        else
        {
            Book *tmp = this->head;
            while (tmp ->next != NULL)
            {
                tmp = tmp ->next;
            }
            tmp->next = new_book;
        }
    }
}

bool Library::Is_Empty () const
{
    return this->head == NULL;
}
bool Library::Is_Full () const
{
    Library * newacc;
    try
    {
        newacc = new Library;
        delete newacc;
        return false;
    }
    catch (bad_alloc e)
    {
        return true;
    }
}
bool Library::search_id (int id) const
```

```
{
        Book *tmp = this->head;
        while (tmp != NULL)
        {
                if (id == tmp->id)
                {
                        return true;
                }
                tmp = tmp ->next;
        }
        return false;
}
void Library::display_all() const
{
        Book * tmp = head;
        while (tmp != NULL)
        {
        cout << tmp->id << "  "<< tmp ->title << " " << tmp ->author
                << " " << tmp->number_pages
                        << " "  << tmp ->isborrow << endl;
                tmp = tmp->next;
        }
}
void Library::display_book(int id) const
{
        Book * tmp;
        if (this->search_id(id) == true)
        {
                tmp = this->head;
                while (tmp != NULL)
                {
                        if (id == tmp->id)
                        {
        cout << tmp->id << "  " << tmp->title << "  "
                << tmp ->isborrow << tmp->author
                                        << tmp ->number_pages << endl;
                                break;
                        }
```

```
				tmp = tmp ->next ;
			}
		}
		else
		{
			cout << id <<" not found" << endl;
		}
}
void Library::delete_book(int id)
{
		if (this ->search_id (id))
		{
			Book * tmp = head,*t;
			while (tmp != NULL)
			{
				if (tmp ->id == id)
					break;
				t = tmp;
				tmp = tmp ->next;
			}
			if (tmp  == head)
			{
				head = head ->next;
				delete tmp;
			}
			else
			{
				if (tmp != NULL)
				{
					t ->next = tmp ->next;
					delete tmp;
				}
				else
				{
					cout << id <<" Not Found" << endl;
				}
			}
		}
```

```
        else
        {
                cout << id << " not found" << endl;
        }
}
void Library::search_title(string x ) const
{
        Book * tmp = this->head ;
        while (tmp != NULL)
        {
                if (x.compare (tmp->title) == 0)
                {
                        this->display_book(tmp->id);
                }
                tmp = tmp ->next;
        }
}
void Library::search_author(string x)const
{
Book * tmp = this->head ;
        while (tmp != NULL)
        {
                if (x.compare (tmp->author) == 0)
                {
                        this->display_book(tmp->id);
                }
                tmp = tmp ->next;
        }
}
void Library::borrow_book(int id) const
{
        if (this->search_id(id) )
        {
                Book *t = head;
                while (t != NULL)
                {
                        if (t->id == id)
                        {
```

```
                    if (t->isborrow == 1)
                    {
                cout << t->id <<" is already borrow" << endl;
                    }
                    else
                    {
                        t->isborrow = 1;
                        cout << t->id <<" is borrow now" << endl;
                    }
                    break;
                }
            }
        }
        else
        {
            cout << id << "not Found" << endl;
        }
}
void Library::return_book (int id) const
{
        if (this->search_id (id))
        {
            Book *tmp = head;
            while (tmp != NULL)
            {
                if (id == tmp ->id)
                {
                    if (tmp ->isborrow == 1)
                    {
                        tmp ->isborrow = 0;
                        cout << id <<" is returned" << endl;
                    }
                    else
                    {
                        cout << id << " not borrow" << endl;
                    }
                    break;
                }
                tmp = tmp ->next;
            }
        }
        else
        {
            cout << id <<" Not Found" << endl;
        }
}
int main()
{
        int choice,id;
        string title,author;
        Library *b = new Library;
        while (true)
        {
            cout << "Enter your choice : " <<endl;
            cin >> choice;
            switch (choice)
            {
            case 0:
                cout << "All books : "<< endl;
                b->display_all();
                system("pause");
              exit(0);
                break;
            case 1:
                b->add_book();
                break;
            case 2:
                cout << "Enter id : " << endl;
                cin >> id;
                b->delete_book(id);
                break;
            case 3:
                cout << "Enter search id : " << endl;
                cin >> id;
                b ->display_book(id);
                break;
            case 4:
                    cout << "Enter title :" << endl;
                    cin >> title;
                b->search_title(title);
                    break;
            case 5:
                    cout << "Enter author :" << endl;
                    cin >> author;
                b->search_author(author);
                    break;
            case 6:
                    b ->display_all();
                    break;
            case 7:
                    cout << "Enter id of book :" << endl;
                    cin >> id;
                    b->borrow_book(id);
                    break;
            case 8:
                    cout << "Enter id of book :" << endl;
                    cin >> id;
                    b->return_book(id);
                    break;
            default :
                    cout << "try again " << endl;
                    break;
            }
        }
        system("pause");
        return 0;
}
```

**<u>To-Do List:</u>**

**Creating a to-do list that uses dynamic data structures. Implement functionalities like Adding a task, Removing a task, Displaying the to-do list, and check if the task is done or not.**

```
struct To_Do
{
	int id;
	string name;
	bool status;
};
```

**<u>Hospital Management System:</u>**

**Create a simple hospital management system that uses dynamic data structures to manage patients information like ID, name, age, contact details, etc. Implement functionalities like adding new patient, searching for existing patient, retrieve patient information, update and delete patient.**

```
struct Patient
{
	int id;
	string name,contact;
	double age;
	Patient *next;
};
```