EAST WEST UNIVERSITY

REPORT

On

Inventory Management System

Submitted To

- Tanni Mittra
  Senior Lecturer
  Department of Computer Science and Engineering

Submitted By

- Name of student: Md. Azharul Islam.
  Student id: 2020-2-60-019.
- Name of student: Habibur Rahman.
  Student id: 2020-2-60-124.
- Name of student: Rubiatis Sadia.
  Student id: 2020-2-60-155.

Department of Computer Science and Engineering
Course Code: CSE207.

Date: 22 January, 2022

➢ Introduction

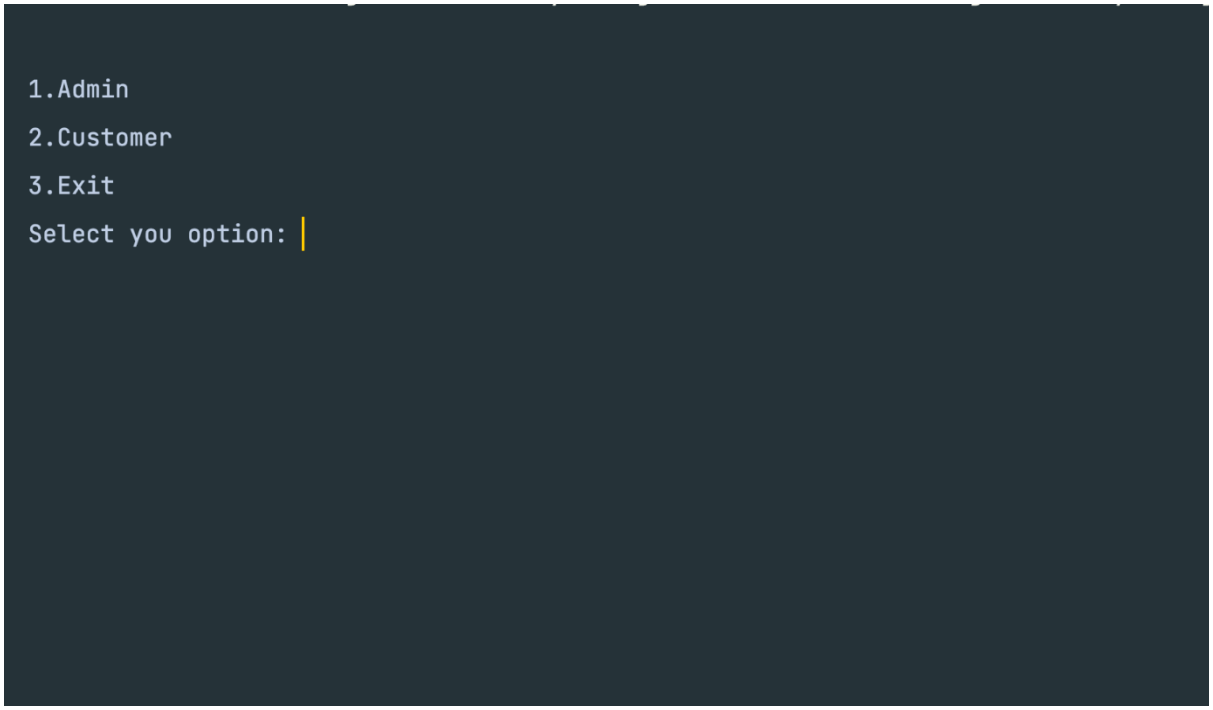
There is an online shop that contains many kinds of electronic tools and they sell their product depending on the customers' request. Each client gives an order to buy a tool. The shop employees will search for the required tools upon customers' request in First Come First Served basis.

2. System Requirements: (Recommended) Processor: Intel Core i3-4130 dual core / AMD Athlon 600, RAM: 128 mb, Operating System: Windows 7/8/10, IDE: NetBeans IDE 8.2 or higher.

3.System Design: As we did the project with the source code only there is no Graphical User Interface was used when developed. No Database was used. (OPTIONAL: For making it more efficient we collected some methods to design it better in future) In our project at first, we have used DLL for storing categories and products. Next, we have used Queue for storing customers' orders information because we need to serve customers First Come First Served basis. After that, we have used Stack for storing the information of sold tools of per category. Finally, we have built report that will show monthly selling reports of tools and showed the last five sold report.

Project's Screenshots

Main Menu



Admin Page

```
1.Admin
2.Customer
3.Exit
Select you option: 1

1.Create Category
2.Insert Item
3.Display All Item
4.Order List
5.Last sold Items
6.Monthly Sell Data
7.Go to Main
Select you option:
```

Customer Page

```
1.Admin
2.Customer
3.Exit
Select you option: 2

================================== ALL ITEMS ======================================

-------------------------------------------------------------------------------
Category               ID             Name                   Price          Qty
-------------------------------------------------------------------------------

CTG8579 | Hand Tools        pd01               Screwdriver            200            20

-------------------------------------------------------------------------------

Do want place an order? y/n --> y
Enter Product ID: pd01
Enter Quantity: 2
You ordered for: Screwdriver
Do you want to order again ? y/n-->n

=================== Invoice ======================
Purchase Date: 22/1/2022
--------------------------------------------------
Item Name       Price   Quantity
--------------------------------------------------
Screwdriver     200     2
--------------------------------------------------
                Total: $400
================ Happy Shopping! ====================
```

Project Codes:

Main.cpp

```
#include <iostream>

using namespace std;

#include "List.h"

#include "Product.h"

#include "Order.h"

#include "Queue.h"

#include "Stack.h"

#include <string>



//All List declared globally

List* list  = new List();

Queue* queue = new Queue();

Stack* stack = new Stack();


//function Prototype

string randID();

void adminMenu();
```

```
void mainMenu();


//fun for creating order
Order createAnOrder(Order order){
    string id;
    int qty;
    cout << "Enter Product ID: ";
    cin >> id;
    Product pd = list->searchByProductID(id);
    cout << "Enter Quantity: ";
    cin >> qty;
    cout << "You ordered for: " << pd.getName() << endl;
    pd.setQty(qty);
    order.addOrderedProduct(pd);
   list->decreaseQty(id,qty);


    //storing ordered product in stack.
    stack->push(pd);
    return order;
}
```

```
//fun for Placing order
void placeAnOrder(){
    char op,opt ='y';
    list->displayAllProduct();
    cout << "Do want place an order? y/n --> ";
    cin >> op;
    if (tolower(op) =='y'){
        Order order = Order();
        order.setOrderId(randID());
        order = createAnOrder(order);
        while (tolower(opt) =='y'){
            cout << "Do you want to order again ? y/n-->";
            cin >> opt;
            if (tolower(opt)=='y')
                order = createAnOrder(order);
        }
        //show invoice
        order.getInvoice();
        //insert order customer's order information in Queue.
        queue->enquey(order);
```

```
    }
}

//Main function
int main() {

    list->createCategory("Electronics");
    mainMenu();

    return 0;
}



//Main Menu
void mainMenu(){
    int option;
    bool stop = false;

    do {

        cout
```

```
            << endl <<"1.Admin" << endl
            << "2.Customer" << endl
            << "3.Exit" << endl;

    cout << "Select you option: ";
    cin >> option;

    switch (option) {
        case 1:
            adminMenu();
            break;
        case 2:
            placeAnOrder();
            break;
        case 3:
            stop = true;
            break;

    }

} while (!stop);
```

```
}

//Admin menu
 void adminMenu() {

    string catName;
    int option;
    bool stop = false;

    do {
        cout
                << "\n1.Create Category" << endl
                << "2.Insert Item" << endl
                << "3.Display All Item" << endl
                << "4.Order List" << endl
                << "5.Last sold Items" << endl
                << "6.Monthly Sell Data" << endl
                << "7.Go to Main" << endl;
```

```
cout << "Select you option: ";
cin >> option;

switch (option) {
    case 1:
        list->displayCategory();
        cout << "Enter Category Name: ";
        cin>>catName;
        list->createCategory(catName);
        break;
    case 2:
        list->insertProduct();
        break;
    case 3:
        list->displayAllProduct();
        break;
    case 4:
        queue->displayOrder();
        break;
    case 5:
```

```
            stack->displayLastFiveSell();
            break;
        case 6:
            stack->displayMonthlyData();
            break;
        case 7:
            stop = true;
            mainMenu();
            break;
    }

  } while (!stop);

}

//Random ID Generator
string randID() {

  string id = "";
  // srand(time(NULL));
  int random = rand()%9999;
```

```
    id ="OR"+to_string(random);
    return id;
}
```

Product.h

```
#ifndef INVENTORY_MANAGEMENT_LIST_H
#define INVENTORY_MANAGEMENT_LIST_H

#include <iostream>
using namespace std;
#include <ctime>
#include "Product.h"
#include <algorithm>

class PDNode{

public:

    Product product;
    PDNode* pdNext;
```

```
    PDNode(Product product){
      this->product = product;
      pdNext = NULL;
    }

};

class CTGNode{

public:
    string ctgName;
    string id;
    CTGNode* prev;
    PDNode* next;

    CTGNode(string ctgName){

      this->ctgName = ctgName;
      this->id = randID();
      this->prev = NULL;
```

```
        this->next = NULL;
    }

    const string &getId() const {
        return id;
    }

    void setId(const string &id) {
        CTGNode::id = id;
    }

    string randID() {

        string id = "";
        srand(time(NULL));
        int random = rand()%9999;
        id ="CTG"+to_string(random);
        return id;
    }

};
```

```
class List {
    CTGNode * head;
public:

    List(){
        this->head = NULL;
    }

    //Create a category to insert product
    void createCategory(string name){

        CTGNode* newCTGNode = new CTGNode(name);

        if(head == NULL){
            head = newCTGNode;
        } else {

            CTGNode* temp = head, *end = head;

            while (temp!= NULL)
```

```
        {
            end = temp;
            temp = temp->prev;
        }
        end->prev = newCTGNode;
    }
}

//insert product after creating category if is not created before.
void insertProduct() {

    if(head == NULL){
        cout << "No category found. Please create a category first.";
    } else {

        displayCategory();
        string  catID;
        //Searching Category
        cout <<"Enter Category ID: ";
         cin>> catID;
        CTGNode *temp = head;
```

```
while (temp!=NULL)
{
    if (temp->id == stringToUpper(catID))
    {
        Product product = createProduct();
        PDNode* newPDNode = new PDNode(product);
        PDNode* head1 = temp->next;
        if(head1 == NULL){
            temp->next = newPDNode;
        } else {

            PDNode* temp1 = head1, *end;

            while (temp1 != NULL)
            {
                end = temp1;
                temp1 = temp1->pdNext;
            }
            end->pdNext = newPDNode;
        }
```

```
            }
            temp = temp->prev;
        }
    }
}

//creating product method for code simplicity
Product createProduct(){

    string name;
    string id;
    double price;
    int qty;

    Product product =  Product();

    cout << "Enter Product ID: ";
    cin >> id;
    product.setId(id);
    cout << "Enter Product Name: ";
    cin >> name;
```

```
        product.setName(name);
        cout << "Enter Price: ";
        cin >> price;
        product.setPrice(price);
        cout << "Enter Quantity: ";
        cin >> qty;
        product.setQty(qty);

        return product;
    }

    //display available category in the list.
    void displayCategory(){

        CTGNode *nextNode = head;

        cout << "Category: \n";
        while (nextNode!= NULL)
        {
            cout <<"" << nextNode->id <<" | Name: "<<nextNode->ctgName
<<endl;
```

```
            nextNode = nextNode->prev;
        }
        cout << endl;
    }

    //display all product by category.
    void displayAllProduct() {

        CTGNode *nextNode = head;
        PDNode *pdHead = head->next;
        cout << "\n=================================== ALL
ITEMS ====================================" << endl;
        cout << "\n--------------------------------------------------------------------------
---------" << endl;
        cout << "Category\t\t\tID\t\t\tName\t\t\t\tPrice\t\t\tQty";
        cout << "\n--------------------------------------------------------------------------
---------" << endl;
        while (nextNode != NULL)
        {
            //product print

                while (pdHead != NULL) {
```

```
                cout << "\n" << nextNode->id <<" | "<< nextNode->ctgName;
                cout <<"\t\t"<< pdHead->product.getId()
                 << "\t\t\t\t"<<pdHead->product.getName()
                << "\t\t\t\t"<<pdHead->product.getPrice()
                 << "\t\t\t\t"<<pdHead->product.getQty() <<endl;


                pdHead = pdHead->pdNext;
            }
                //next category
            nextNode = nextNode->prev;
                if(nextNode!=NULL)
                    pdHead = nextNode->next; //next product
          cout << "\n-------------------------------------------------------------------------
-----------" << endl;
        }
        cout << endl;

    }

    //search product by id
```

```
Product searchByProductID(string id){

    CTGNode *nextNode = head;
    PDNode *pdHead = head->next;

    while (nextNode != NULL)
    {

        while (pdHead != NULL) {

            if(pdHead->product.getId()==id)
                return pdHead->product;



           pdHead = pdHead->pdNext;
        }
        nextNode = nextNode->prev;
        if(nextNode!=NULL)
            pdHead = nextNode->next;
    }
```

```
}

void  decreaseQty(string id, int qty) {

    int existQTy;
    CTGNode *nextNode = head;
    PDNode *pdHead = head->next;

    while (nextNode != NULL)
    {

        while (pdHead != NULL) {

            if(pdHead->product.getId()==id) {
                existQTy = pdHead->product.getQty()-qty;
                pdHead->product.setQty(existQTy);
            }
            pdHead = pdHead->pdNext;
        }
        nextNode = nextNode->prev;
        if(nextNode!=NULL)
```

```
            pdHead = nextNode->next;
        }
    }

    //make string upper
    string stringToUpper(string s){
        transform(s.begin(),s.end(), s.begin(),::toupper);
        return s;
    }
};


#endif //INVENTORY_MANAGEMENT_LIST_H
```

List.h

```
#ifndef INVENTORY_MANAGEMENT_LIST_H
#define INVENTORY_MANAGEMENT_LIST_H

#include <iostream>
using namespace std;
```

```
#include <ctime>
#include "Product.h"
#include <algorithm>

class PDNode{

public:

    Product product;
    PDNode* pdNext;

      PDNode(Product product){
        this->product = product;
        pdNext = NULL;
    }

};

class CTGNode{

public:
```

```
string ctgName;
string id;
CTGNode* prev;
PDNode* next;

CTGNode(string ctgName){

    this->ctgName = ctgName;
    this->id = randID();
    this->prev = NULL;
    this->next = NULL;
}

const string &getId() const {
    return id;
}

void setId(const string &id) {
    CTGNode::id = id;
}
```

```
    string randID() {

        string id = "";
        srand(time(NULL));
        int random = rand()%9999;
        id ="CTG"+to_string(random);
        return id;
    }

};

class List {
    CTGNode * head;
public:

    List(){
        this->head = NULL;
    }

    //Create a category to insert product
    void createCategory(string name){
```

```
    CTGNode* newCTGNode = new CTGNode(name);

    if(head == NULL){
        head = newCTGNode;
    } else {

        CTGNode* temp = head, *end = head;

        while (temp!= NULL)
        {
            end = temp;
            temp = temp->prev;
        }
        end->prev = newCTGNode;
    }
}

//insert product after creating category if is not created before.
void insertProduct() {
```

```
if(head == NULL){
    cout << "No category found. Please create a category first.";
} else {


    displayCategory();
    string  catID;
    //Searching Category
    cout <<"Enter Category ID: ";
     cin>> catID;
    CTGNode *temp = head;


    while (temp!=NULL)
    {
        if (temp->id == stringToUpper(catID))
        {
            Product product = createProduct();
            PDNode* newPDNode = new PDNode(product);
            PDNode* head1 = temp->next;
            if(head1 == NULL){
                temp->next = newPDNode;
            } else {
```

```
                PDNode* temp1 = head1, *end;


                while (temp1 != NULL)
                {
                    end = temp1;
                    temp1 = temp1->pdNext;
                }
                end->pdNext = newPDNode;
            }
        }
        temp = temp->prev;
    }
  }
}


//creating product method for code simplicity
Product createProduct(){


   string name;
   string id;
```

```
    double price;
    int qty;

    Product product =  Product();

    cout << "Enter Product ID: ";
    cin >> id;
    product.setId(id);
    cout << "Enter Product Name: ";
    cin >> name;
    product.setName(name);
    cout << "Enter Price: ";
    cin >> price;
    product.setPrice(price);
    cout << "Enter Quantity: ";
    cin >> qty;
    product.setQty(qty);

    return product;
}
```

```
    //display available category in the list.
    void displayCategory(){

        CTGNode *nextNode = head;

        cout << "Category: \n";
        while (nextNode!= NULL)
        {
            cout <<"" << nextNode->id <<" | Name: "<<nextNode->ctgName
<<endl;
            nextNode = nextNode->prev;
        }
        cout << endl;
    }

    //display all product by category.
    void displayAllProduct() {

       CTGNode *nextNode = head;
       PDNode *pdHead = head->next;
        cout << "\n==================================== ALL
ITEMS ====================================" << endl;
```

```
        cout << "\n-------------------------------------------------------------------------
---------" << endl;

        cout << "Category\t\t\t\tID\t\t\t\tName\t\t\t\t\tPrice\t\t\tQty";

        cout << "\n-------------------------------------------------------------------------
---------" << endl;

       while (nextNode != NULL)

       {

          //product print


              while (pdHead != NULL) {

                 cout << "\n" << nextNode->id <<" | "<< nextNode->ctgName;

                 cout <<"\t\t"<< pdHead->product.getId()

                  << "\t\t\t\t"<<pdHead->product.getName()

                 << "\t\t\t\t"<<pdHead->product.getPrice()

                  << "\t\t\t\t"<<pdHead->product.getQty() <<endl;



                 pdHead = pdHead->pdNext;

            }

              //next category

           nextNode = nextNode->prev;

              if(nextNode!=NULL)
```

```
                pdHead = nextNode->next; //next product

        cout << "\n----------------------------------------------------------------------
-----------" << endl;

    }

    cout << endl;


}


//search product by id

Product searchByProductID(string id){


    CTGNode *nextNode = head;

    PDNode *pdHead = head->next;


    while (nextNode != NULL)

    {


        while (pdHead != NULL) {


            if(pdHead->product.getId()==id)

                return pdHead->product;
```

```
            pdHead = pdHead->pdNext;
        }
        nextNode = nextNode->prev;
        if(nextNode!=NULL)
            pdHead = nextNode->next;
    }

}

void  decreaseQty(string id, int qty) {

    int existQTy;
    CTGNode *nextNode = head;
    PDNode *pdHead = head->next;

    while (nextNode != NULL)
    {

        while (pdHead != NULL) {
```

```
            if(pdHead->product.getId()==id) {
                existQTy = pdHead->product.getQty()-qty;
                pdHead->product.setQty(existQTy);
            }
            pdHead = pdHead->pdNext;
        }
        nextNode = nextNode->prev;
        if(nextNode!=NULL)
            pdHead = nextNode->next;
    }
}

//make string upper
string stringToUpper(string s){
    transform(s.begin(),s.end(), s.begin(),::toupper);
    return s;
}
};
#endif //INVENTORY_MANAGEMENT_LIST_H
```

Queue

```
#ifndef INVENTORY_MANAGEMENT_QUEUE_H
#define INVENTORY_MANAGEMENT_QUEUE_H

#include <iostream>
#include "Order.h"
using namespace std;

class Node {
public:

    Order order;
    Node *next;

    Node(Order order){
        this->order = order;
        next = NULL;
    }

};
```

```
class Queue {

    public:
        Node * front;
        Node* rear;

        Queue(){
            front = NULL;
            rear = NULL;
        }

        bool isEmpty(){

            if(front == NULL && rear == NULL)
                return true;
            else
                return false;
        }
```

```
void enquey(Order val){

    Node *newNode = new Node(val);

    if (front == NULL)
    {
        front = newNode;
        rear = newNode;

    } else {
        rear->next = newNode; //adding node to rear next or in the end.
        rear = newNode; //Updating rear.
    }
}

Order dequey(){

    if(isEmpty())
        cout<<"Underflow";
    else{
        Order dt = Order();
```

```
        //only one element/node in queue.
        if(front == rear)
        {
            dt = front->order;
            delete(front);
            front = rear = NULL;
            return dt;
        }
        else
        {   dt = front->order;
            Node *temp = front;
            front = front->next;
            delete(temp);
            return  dt;
        }
    }
}

int size(){
    int count = 0;
    if(!isEmpty()){
```

```
        Node* temp = front;


        while (temp !=NULL)
        {   count++;
            temp = temp->next;
        }
        return count;
    }
    else
        return count;
}

void displayOrder() {

    if (isEmpty())
        cout<<"No Order!\n";
    else
    {
        Node *temp = front;
        while( temp !=NULL)
```

```
        {
            cout<<temp->order.getOrderId()<<" ";
            temp= temp->next;
        }
        cout << endl;
    }
}

Queue* CopyQueue() {

    Queue *q2 = new Queue();

    while(front!=NULL)
    {

        Order data = front->order;
        q2->enquey(data);
        front = front->next;
    }
    return q2;
}
```

```
    Node* peek(){
        return front;
    }

    Order getFront(){
        if(!isEmpty())
            return (front->order);
    }


};


#endif //INVENTORY_MANAGEMENT_QUEUE_H
```

Order.h

```
#ifndef INVENTORY_MANAGEMENT_ORDER_H
#define INVENTORY_MANAGEMENT_ORDER_H
#include "Product.h"
```

```
#include <string>
#include <vector>
#include <ctime>
using namespace std;

class Order{
    string orderID;
    vector<Product> item;
    double bill;
    time_t now;
    tm *purchaseDate;
    bool isDelivered;

public:

    Order(){
        this->bill = 0.0;
        now = time(0);
        purchaseDate = localtime(&now);
    }
```

```
tm *getPurchaseDate() const {
    return purchaseDate;
}

const vector<Product> &getItem() const {
    return item;
}

void addOrderedProduct(Product product){
   item.push_back(product);
}


const string &getOrderId() const {
    return orderID;
}

void setOrderId(const string &orderId) {
    orderID = orderId;
}
```

```
void setBill(double bill) {
    Order::bill = bill;
}

bool isDelivered1() const {
    return isDelivered;
}

void setIsDelivered(bool isDelivered) {
    Order::isDelivered = isDelivered;
}

double getBill(){

    for(int i =0; i< item.capacity(); i++){
        bill += (item[i].getPrice()*item[i].getQty());
    }
    return bill;
}

void getInvoice(){
```

```
        cout << "\n==================== Invoice
======================\n";

        cout << "Purchase Date: "

        << purchaseDate->tm_mday

        <<"/" <<purchaseDate->tm_mon+1 <<

        "/"<<purchaseDate->tm_year+1900 << endl;

        cout << "-------------------------------------------------\n";

        cout << "Item Name\t\tPrice\tQuantity"<< endl;

        cout << "-------------------------------------------------\n";

        for(int i =0; i<item.capacity(); i++)

            cout <<item[i].getName() << "\t\t" << item[i].getPrice()  <<"\t\t" <<
item[i].getQty() << endl;

        cout << "-------------------------------------------------\n";

        cout << "\t\t\t\tTotal: $" << getBill() <<endl;

        cout << "================ Happy Shopping!
====================\n";

    }

}; #endif //INVENTORY_MANAGEMENT_ORDER_H
```

Stack.h

```
#ifndef INVENTORY_MANAGEMENT_STACK_H
#define INVENTORY_MANAGEMENT_STACK_H

#include <iostream>
using namespace std;
#include "Product.h"

class Node1 {

public:

    Product data;
    Node1 *next;

    Node1(Product data){
        this->data = data;
        next = NULL;
    }
```

```
};

class Stack {

public:
    Node1 * top;

    Stack(){
        top = NULL;
    }

    void push(Product val){
        Node1 *newNode = new Node1(val);


        if(top == NULL){
            top = newNode;
        } else {
            newNode->next = top;
            top = newNode;
```

```
    }


}

//remove from stack
Product pop(){


    if (top == NULL)
    {
        cout<<"Underflow"<<endl;
    } else {
        Product topData = top->data;
        Node1 *temp = top;
        top = top->next;
        delete(temp);
        return topData;
    }


}

//Get stack top data
```

```
Product StackTop(){
    if(!isEmpty()) {

        return top->data;
    } else
        cout << "Underflow ";

}

//empty or have data
bool isEmpty(){
    if(top==NULL) return true;
    else return false;

}

//display all in stack
void display(){

    Node1 *nextNode = top;
    while (nextNode!= NULL)
```

```
    {
        cout << nextNode->data.getId() <<" | ";
        nextNode = nextNode->next;
    }
    cout << endl;
}

//search product by order Month and Year.
Product searchByMonthYr(int month, int year)
{
    Node1 *nextNode = top;
    while (nextNode!= NULL)
    {
        if (nextNode->data.getSellingDate()->tm_mon == month && nextNode->data.getSellingDate()->tm_year+1900 == year)
            return nextNode->data;


        nextNode = nextNode->next;
    }
    cout << endl;
```

```
}

//print function for month
void printMonth(int i){



        switch (i) {


            case 0:
                cout << "==================================== January 2022  ====================================" << endl;
                cout <<"\n-------------------------------------------------------------------------------------"<< endl;
                cout <<"\t\tDate\t\t\tID\t\t\tName\t\t\t\tPrice\t\tQuantity";
                cout <<"\n-------------------------------------------------------------------------------------"<< endl;
                break;
            case 1:
                cout << "========= February 2022  ===========" << endl;
                cout <<"\n----------------------------"<< endl;
                cout <<"Sr.\tID\t\tName";
                cout <<"\n----------------------------"<< endl;
```

```
    break;
case 2:
    cout << "========= March 2022  ==========" << endl;
    cout <<"\n----------------------------"<< endl;
    cout <<"Sr.\tID\t\tName";
    cout <<"\n----------------------------"<< endl;
    break;
case 3:
    cout << "========= April 2022  ==========\n" << endl;
    cout <<"\n----------------------------"<< endl;
    cout <<"Sr.\tID\t\tName";
    cout <<"\n----------------------------"<< endl;
    break;
case 4:
    cout << "========= May 2022  ==========\n" << endl;
    cout <<"\n----------------------------"<< endl;
    cout <<"Sr.\tID\t\tName";
    cout <<"\n----------------------------"<< endl;
    break;
case 5:
    cout << "========= June 2022  ==========\n" << endl;
```

```
    cout <<"\n----------------------------"<< endl;
    cout <<"Sr.\tID\t\tName";
    cout <<"\n----------------------------"<< endl;
    break;
case 6:
    cout << "========= July 2022  ===========\n" << endl;
    cout <<"\n----------------------------"<< endl;
    cout <<"Sr.\tID\t\tName";
    cout <<"\n----------------------------"<< endl;
    break;
case 7:
    cout << "========= August 2022  ===========\n" << endl;
    cout <<"\n----------------------------"<< endl;
    cout <<"Sr.\tID\t\tName";
    cout <<"\n----------------------------"<< endl;
    break;
case 8:
    cout << "========= September 2022  ===========\n" << endl;
    cout <<"\n----------------------------"<< endl;
    cout <<"Sr.\tID\t\tName";
    cout <<"\n----------------------------"<< endl;
```

```
            break;
        case 9:
            cout << "========= October 2022  ==========\n" << endl;
            cout <<"\n---------------------------"<< endl;
            cout <<"Sr.\tID\t\tName";
            cout <<"\n---------------------------"<< endl;
            break;
        case 10:
            cout << "========= November 2022  ==========\n" << endl;
            cout <<"\n---------------------------"<< endl;
            cout <<"Sr.\tID\t\tName";
            cout <<"\n---------------------------"<< endl;
            break;
        case 11:
            cout << "========= December 2022  ==========\n" << endl;
            cout <<"\n---------------------------"<< endl;
            cout <<"Sr.\tID\t\tName";
            cout <<"\n---------------------------"<< endl;
            break;

}
```

```
}

//print monthly Sell product
void displayMonthlyData() {
    Node1 *nextNode = top;
    double total = 0.0;
    bool flag = true;

    while (nextNode != NULL) {
        int month = nextNode->data.getSellingDate()->tm_mon;
        int year = (nextNode->data.getSellingDate()->tm_year + 1900);

        if (month == 0 && year == 2022) {
            if (flag) {
                flag = false;
                printMonth(month);
            }
            cout
                << "\t\t" << nextNode->data.getSellingDate()->tm_mday
               <<"/" <<nextNode->data.getSellingDate()->tm_mon+1 <<
                "/"<<nextNode->data.getSellingDate()->tm_year+1900
```

```
                << "\t\t" << nextNode->data.getId()
                 << "\t\t\t" << nextNode->data.getName()
                 << "\t\t\t" << nextNode->data.getPrice()*nextNode->data.getQty()
                 << "\t\t\t" << nextNode->data.getQty() <<endl;

            total += nextNode->data.getPrice()*nextNode->data.getQty();

            nextNode = nextNode->next;
          }
        }
        cout <<"\n-------------------------------------------------------------------------------
----------"<< endl;
        cout << "\t\t\t\t\t\t\tTotal Sell on this month: $" << total << endl;
    }

    //print last 5 sell
    void displayLastFiveSell(){

        int count = 1;
        Node1 *nextNode = top;
        cout <<"\n----------------------------"<< endl;
```

```
    cout <<"\tID\t\tName";
    cout <<"\n----------------------------"<< endl;

    while (nextNode!= NULL)
    {
        if(count<=5) {
            cout << count << ". \t" << nextNode->data.getId()
                << "\t\t" << nextNode->data.getName() <<endl;
            count++;
            nextNode = nextNode->next;
        } else
            break;
    }

}

//total selling item
int size(){
    int count = 0;
    if(top!=NULL){
```

```
        Node1 *temp = top;
        while (temp!=NULL)
        {
            count++;
            temp = temp->next;
        }
    }
    return count;
  }
};


#endif //INVENTORY_MANAGEMENT_STACK_H
```

Conclusion

In this project we have implemented the basic usage of some Data Structures' concepts and built a software of the project "Inventory Management System". By doing this project we have learned basic Data Structure's concepts.