PROJECT REPORT

OF

" PROFIT EARNED IN SHARE MARKET "

Delhi **T**echnological **U**niversity

Submitted in the partial fulfillment of the Degree of Bachelor of Technology
In
Software Engineering

SUBMITTED BY :-
Shubham (2K19/SE/119)

PROFIT EARNED IN SHARE MARKET

ACKNOWLEDGMENT

First and the foremost I would like to thank to my almighty for giving me courage.

A word of thanks to my teacher, friends and other sources that gave an unending support and helped me in numerous ways from the first stage of my term assignment conceived.

I would also like to thank my family members for their whole hearted support and cooperation.

I duly acknowledge the contribution of Mr. Rahul for invaluable help. Making this project is an uphill task and would have not been possible without proper and timely assistance of Mr. Rahul.

I would also thanks to all my friends for forwarding their suggestions to make necessary modifications.

Special thanks to Mr. Rahul for his able guidance in this project.

TABLE OF CONTENTS

OBJECTIVE

We need to find maximum profit that can be earned by buying and selling shares of one particular product at most k times with a constraint that a new transaction can only start after previous transaction is complete, i.e. we can only hold at most one share at a time.

INTRODUCTION

If anyone considered investing their money to grow wealth outside super. If that's the case, maybe he/she have thought about shares. Or maybe he/she have been watching the share market closely and have been wondering whether now's the time to buy. May be volality concerns him/her. A share is a piece of company, issued by a company to raise money that's often called 'capital'. We can buy shares directly from a company float (or new share sale) by filling out a prospectus and sending in our money to the company or via a managed fund that includes many different shares selected by an experienced share buyer.

The price of shares is based on how much the company is estimated to be worth, how many shares have been issued. If the company business is profitable, we may receive a share of those profits as a payment called a dividend. A share market

also called a stock exchange, is where shares are bought and sold. The share market is made up of thousands of individual companies. Many are household names and make the products or services that we use every day, like phones, cars and groceries.

LESSON LEARNT

1. Simulation program with C++
2. Share market behaviour
3. Dynamic programming
4. About Stock Market Simulator
5. Stock Span
6. Data File Handling
7. Entity, Attribute, Activity

SIMULATION PROGRAM WITH C++ :

The problem of choice of programming language and simulation environment which could be most suitable for developing such complex and unconventional models from the viewpoint of their universality, flexibility, effectiveness, and interactive features comes to the fore. There exists specialized ready-made software, such as Maple, Mathematica, MathCAD etc, which provides a range of opportunities sufficient for a number of applications of average complexity, especially for the models consisting of standard subcomponents. On the other hand, a number of domain-specific languages have been developed for the simulation purposes, but C++ usually provides better effectiveness and flexibility.

SHARE MARKET BEHAVIOUR :

The stock market behaves differently at different points of time depending on the overall market trend and so many other factors. The behavior of the stock market or the market trend has different phases that can be categorized in two types – primary trend and the secondary trend. The market behavior is determined by the primary trend of the market. The behavior of the market differs greatly with the market trend. When the market is in a bull phase the market behaves in a certain way and when it is in bearish phase the market behaves differently and of course the correction phase sees the market in different colors. To understand these different behaviors of the stock market we have to understand these market trends and the underlying causes of these market trends.

DYNAMIC PROGRAMMING :

Dynamic programming is a subset of the “divide and conquer” method, given that a solution to a problem depends on the previous solutions obtained from subproblems. The main and major difference between these two methods lies in the superimposition of subproblems in dynamic programming. It can be claimed that the “divide and conquer” method works by following a top-down approach, whereas dynamic programming follows a bottomup approach. A dynamic programming algorithm generally consists of a number of phases that link together to come to the optimal solution: characterization, definition, calculation and construction. Stochastic processes have many applications, particularly for managing transport networks, managing traffic, electric systems, telecommunications, signal processing, filtering, finance, the economy, etc.

STOCK MARKET SIMULATOR :

A stock market simulator is a program or application that attempts to reproduce or duplicate some or all the features of a live stock market on a computer so that a player may practice trading stocks without financial

risk. Paper trading (sometimes also called "virtual stock trading") is a simulated trading process in which would-be investors can 'practice' investing without committing real money.

This is done by the manipulation of imaginary money and investment positions that behave in a manner similar to the real markets. New investors can practice making (or losing) fortunes time and time again before actually committing financially. Investors also use paper trading to test new and different investment strategies. Stock market games are often used for educational purposes.

STOCK SPAN :

The span of the stock's price today is defined as the maximum number of consecutive days (starting from today and going backwards) for which the price of the stock was less than today's price.

For example, if the price of the stock over a period of 7 days are [100, 80, 60, 70, 60, 75, 85], then the stock spans will be [1, 1, 1, 2, 1, 4, 6].

DATA FILE HANDLING :

Files are used to store data in a storage device permanently. File handling provides a mechanism to store the output of a program in a file and to perform various operations on it.

A stream is an abstraction that represents a device on which operations of input and output are performed. A stream can be represented as a source or destination of characters of indefinite length depending on its usage.

In C++ we have a set of file handling methods. These include ifstream, ofstream, and fstream. These classes are derived from fstrembase and from the corresponding iostream class. These classes, designed to manage the disk files, are declared in fstream and therefore we must include fstream and therefore we must include this file in any program that uses files.

In C++, files are mainly dealt by using three classes fstream, ifstream,

ofstream.

- ofstream : This Stream class signifies the output file stream and is applied to create files for writing information to files.
- ifstream : This Stream class signifies the input file stream and is applied for reading information from files.
- fstream : This Stream class can be used for both read and write from/to files.

All the above three classes are derived from fstreambase and from the corresponding iostream class and they are designed specifically to manage disk files. C++ provides us with the following operations in File Handling :

- Creating a file : open()
- Reading data : read()
- Writing new data : write()
- Closing a file : close()

ENTITY : Products (that are sell or buy).
ATTRIBUTE : Money
ACTIVITY : Buy and sell (shares)

GOALS :-

The main goals of this project were to investigate possible investment opportunities, logically weigh their respective risks and benefits, and make educated investment decisions. A detailed understanding of the risks and opportunities for high return presented in each investment opportunity were obtained through extensive research and analysis. This knowledge will aid in the future selection and execution of intelligent investments.

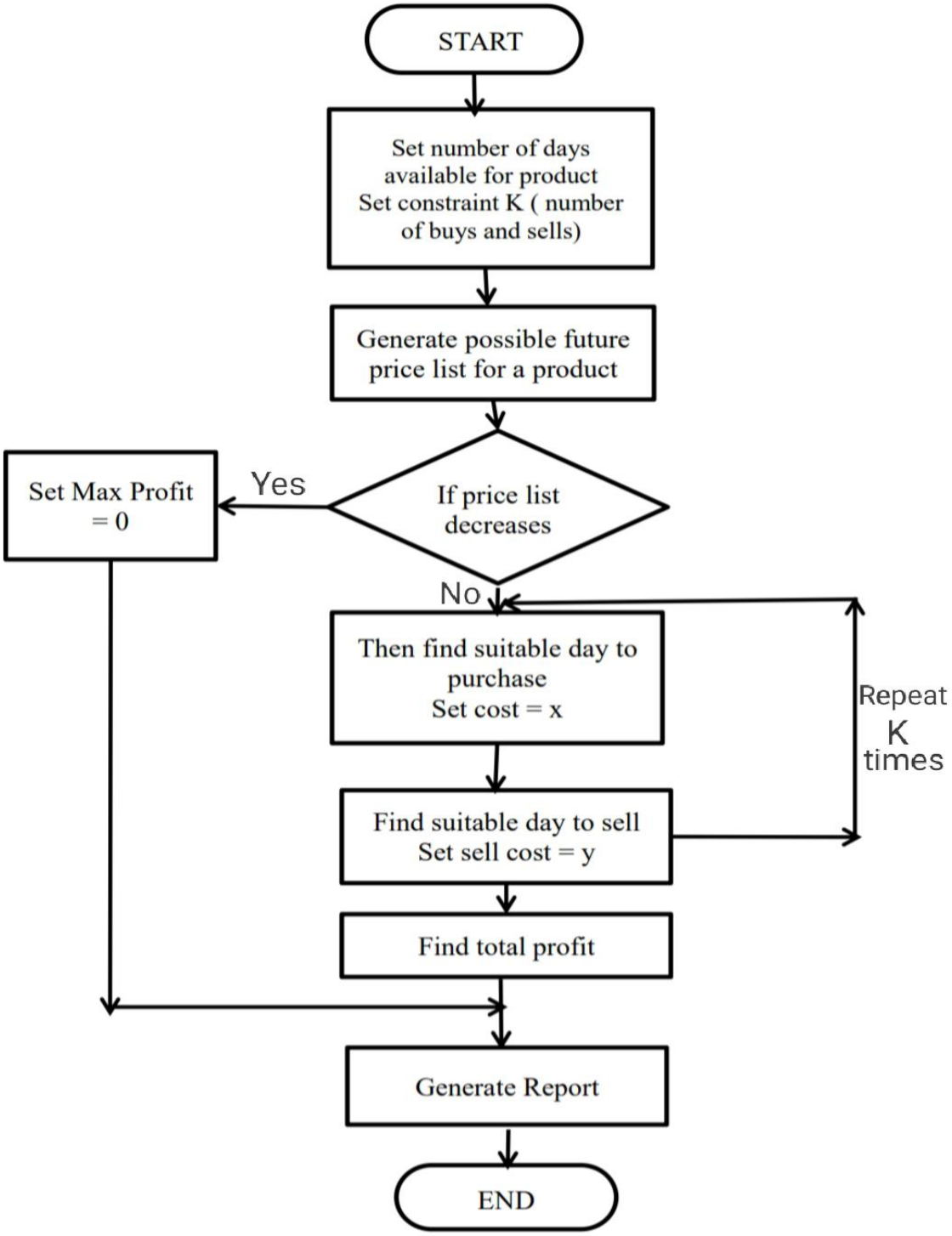
The primary goal of this simulation was to gain a better understanding of personal investment opportunities. The financial goal set for this simulated $100,000 investment was a return of at least $4,000 over the span of 3 weeks. This required an average total gain of at least $200 per day. For this investment simulation a swing - trading scheme was chosen

because of its potential for gain within a short time period. A Day Trading scheme was also employed in order to better understand the subtleties that make it profitable. Other trading strategies were out of the financial scope and timeframe of this simulation.

FEATURES PROVIDED

1. User can add as many accounts as he/she wants.
2. After adding account we will be given by 2500 virtual money.
3. And then we can only purchase shares with max value of our balance.
4. Otherwise insufficient balance message would be shown.
5. Then we need to set number of days for trading.
6. Then our program automatically will generate a price list for a given product in different days with the help of LCG (Linear Congruential Generator).
7. Then we will be asked to select numbers of buys and sells account holder can do.
8. After that our program generates total profit one can earn after final price list.
9. We can delete and add accounts whenever we want.
10. We also provided forgot password routine for account holder.

FLOW CHART :-


START
Set number of days available for product
Set constraint K (number of buys and sells)
Generate possible future price list for a product
If price list decreases
Yes
Set Max Profit = 0
No
Then find suitable day to purchase
Set cost = x
Find suitable day to sell
Set sell cost = y
Repeat K times
Find total profit
Generate Report
END

CODING :

```
/*    C++ program to find out maximum profit by buying and selling a share
atmost k times given stock price of n days.    */

#include <iostream>
#include <fstream>
#include <climits>
#include <stack>
#include <time.h>
using namespace std;

class traderAccount
{
      private :

       string name;
       int age;
       string ID;
       string password;
       int perm;
       float balance;

       public :

      traderAccount()
      {
            perm = 0;
            balance = 2500.00;
      }

      void updateBalance(float bal)
      {
              balance += bal;
      }
```

```
void accountHolder()
{
      cout << "Enter name : ";
        cin >> name;

        cout << "Enter age : ";
        cin >> age;

        cout << "User id : ";
        cin >> ID;

        cout << "Enter password : ";
        cin >> password;
}

int getAge()
{
        return age;
}

string getID()
{
        return ID;
}

void setPassword()
{
        cout << "Entet new password : ";
        cin >> password;
}

string getPassword()
{
        return password;
}

float getBalance()
{
      return balance;
}
```

```
        string getName()
        {
                return name;
        }

        int getPer()
        {
                return perm;
        }

        void permitted()
        {
              perm = 1;
        }
};

void productPriceListGenerator(float price[], int n, traderAccount &c1)
{
        int w;
        srand((unsigned) time(0));
        float randomNumber;

        cout << "\n 1. DEBT CORPORATE BONDS(1050-1705)" << endl;
        cout << " 2. GOLD:Gold ETFs(7050-8028)" << endl;
        cout << " 3. INFRASTRUCTURE INVESTMENT
TRUSTS(INVIT)(3809-4906)" << endl;
        cout << " 4. DABUR INDIA(5106-7000)" << endl;
        cout << " 5. ORIENT ELECTRIC(2104-3078)" << endl;
        cout << " 6. GILLETE INDIA(3400-4500)" << endl;
        cout << " 7. VIP INDUSTRIES(3070-3578)" << endl;
        cout << " 8. COLGATE(1513-3508)" << endl;
        cout << " 9. MARICO(370-508)" << endl;
        cout << "10. GODREJ CONSUMER(685-708)" << endl;
        cout << "11. BAJAJ CONSUMERS(182-250)" << endl;

        int p, N;
        float temp1, temp2;

        do
```

```
{
        cout << "\nSelect product : ";
        cin >> w;

        switch(w)
        {
                case 1:
                p = 1050;
                N = 655;
                break;

                case 2:
                p = 7050;
                N = 978;
                break;

                case 3:
                p = 3809;
                N = 1097;
                break;

                case 4:
                p = 5106;
                N = 1894;
                break;

                 case 5:
                p = 2104;
                N = 974;
                break;

                case 6:
                p = 3400;
                N = 1100;
                break;

                case 7:
                p = 3070;
                N = 505;
                break;
```

```
                case 8:
                p = 1513;
                N = 1995;
                break;

                case 9:
                p = 370;
                N = 138;
                break;

                case 10:
                p = 685;
                N = 23;
                break;

                case 11:
                p = 182;
                N = 68;
                break;
            }

            temp2 = c1.getBalance();
            temp1 = p+N;

            if(temp1 > temp2)
            {
                cout << "Insufficient balance for this product please select
product under " << temp2 << endl;
            }
        }
        while(temp1 > temp2);

        for(int index = 0; index < n; index++)
        {
            randomNumber = (p+rand() % N);
            price[index] = randomNumber;
        }
}

void LCG(int Z, int a, int m, int c, float randArray[], int l)
```

```
{
	int i=0;
	float ui;
	cout << "Product gain/loss in different days :- " << endl;

	for(i=1; i<=l; i++)
	{
		Z = (a*Z+c) % m;
	ui = float(Z)/float(m);

	if(i%2==0 || i%7==0 || i%13==0)
	{
		randArray[i-1] -= ui*15.9763;
		cout << "Day " << i << " :    -" << ui*15.9763;
	}
	else
		{
			randArray[i-1] += ui*157.9763;
			cout << "Day " << i << " :    " << ui*157.9763;
		}
		cout << endl;
	}
}

/*
Stock span is defined as a number of consecutive days prior to the
current day when the price of a stock was less than or equal to the price
at current day.
*/
void stockSpan(float price[], int size)
{
	int *arr = new int[size];
	stack<int> s1;

	int count = 1;
	arr[0] = 1;
	s1.push(0);

	for(int i=1; i<size; i++)
	{
```

```
            while(s1.size() != 0 && price[s1.top()] < price[i])
            {
                  s1.pop();
            }

            if(s1.size() == 0)
            {
                  count = i+1;
            }
            else if(price[s1.top()] >= price[i])
            {
                  count = i - s1.top();
            }

            s1.push(i);
            arr[i] = count;
      }

      for(int i=0; i<size; i++)
      {
            price[i]=arr[i];
      }
}


/*    Function to find out maximum profit by buying & selling a share
atmost k times given stock price of n days    */

float Profit(float price[], int n, int k)
{
      // table to store results of subproblems profit[t][i] stores maximum
      //profit using almost t transactions up to day i (including day i)
      float profit[k + 1][n + 1];

      // For day 0, you can't earn money irrespective of how many times
      //you trade
      for(int i = 0; i <= k; i++)
      {
            profit[i][0] = 0;
      }
```

```
	// profit is 0 if we don't do any transaction (i.e. k =0)
	for(int j = 0; j <= n; j++)
	{
		profit[0][j] = 0;
	}

	// fill the table in bottom-up fashion
	for(int i = 1; i <= k; i++)
	{
		for(int j = 1; j < n; j++)
		{
			float max_so_far = INT_MIN;
			for(int m = 0; m < j; m++)
			{
				max_so_far = max(max_so_far, price[j] - price[m] +
profit[i - 1][m]);
			}

			profit[i][j] = max(profit[i][j - 1],max_so_far);
		}
	}

	return profit[k][n - 1];
}

void insert()
{
	traderAccount c;
	c.accountHolder();
	ofstream w("detail.dat",ios::binary|ios::app);

	if(!w)
	{
		cout << "could not open file";
		exit(0);
	}

	w.write((char*)&c,sizeof(c));
	w.close();
}
```

```
traderAccount& get()
{
	string id, p;
	cout << "Enter ID : ";
	cin >> id;
	cout << "Enter password : ";
	cin >> p;

	traderAccount *c = new traderAccount();
	ifstream r("detail.dat",ios::binary|ios::in);

	while(r.read((char*)&(*c),sizeof(*c)))
	{
		if(id==c->getID()&&p==c->getPassword())
		{
			c->permitted();
			break;
		}
	}

	r.close();
	return *c;
}

void update(traderAccount c1, float bal)
{
	traderAccount c;
	ofstream w("temp.dat",ios::binary);
	ifstream r("detail.dat",ios::binary|ios::in);

	while(r.read((char*)&c,sizeof(c)))
	{
		if(c1.getName()==c.getName() &&
c1.getPassword()==c.getPassword() && c1.getID()==c.getID())
		{
			c.updateBalance(bal);
		}
		w.write((char*)&c,sizeof(c));
	}
```

```
        w.close();
        r.close();
        remove("detail.dat");
        rename("temp.dat","detail.dat");
}

void forgotPassword(traderAccount c1)
{
        traderAccount c;
        string n, id;
        int age;
        cout << "Enter name : ";
        cin >> n;
        cout << "Enter age : ";
        cin >> age;
        cout << "Enter ID : ";
        cin >> id;

        ofstream w("temp.dat",ios::binary);
        ifstream r("detail.dat",ios::binary|ios::in);

        while(r.read((char*)&c,sizeof(c)))
        {

if(c.getName()==c1.getName()&&c.getAge()==c1.getAge()&&c.getID()==
c1.getID())
                {
                        c.setPassword();
                }
                w.write((char*)&c,sizeof(c));
        }

        w.close();
        r.close();
        remove("detail.dat");
        rename("temp.dat","detail.dat");
}
```

```
void show()
{
	traderAccount c;
	ifstream r("detail.dat",ios::binary|ios::in);

	if(!r)
	{
		cout << "File could not open!!!!"<<endl;
	}
	else
	{
		while(r.read((char*)&c,sizeof(c)))
		{
			cout << "Name : " << c.getName() << endl;
			cout << "User ID : " << c.getID() << endl;
			cout << "Balance : " << c.getBalance() << endl;
			cout << endl;
		}
	}
	r.close();
}

void deleteAccount()
{
	traderAccount c;
	string id;
	string p;
	cout << "Enter user id : ";
	cin >> id;

	cout << "Enter password : ";
	cin >> p;

	ifstream r("detail.dat",ios::binary|ios::in);
	ofstream w("temp.dat",ios::binary|ios::app);

	while(r.read((char*)&c,sizeof(c)))
	{
```

```
            if(c.getID()==id && c.getPassword()==p)
            {
                  cout << "account deleted" << endl;
              }
             else
            {
                  w.write((char*)&c,sizeof(c));
            }
      }

      w.close();
      r.close();
      remove("detail.dat");
      rename("temp.dat","detail.dat");
}

// Driver code
int main()
{
      cout << "Welcome to trading world !!" << endl;
      cout << endl;

      char yes;
      traderAccount c1;

      do
      {
            cout << "Do you want to Add an account(y/n) : ";
             cin >> yes;

            if(yes == 'y')
            {
                  insert();
            }
            cout << endl;
      }
      while(yes == 'y');

      traderAccount temp;
      temp = get();
```

```
        int x = temp.getPer();
        if(x)
        {
                cout << "~ Successfully Logged In ~" << endl;
                cout << "Hello " << temp.getName() << endl;
                cout << "Your left balance is : " << temp.getBalance() << endl;

                cout << endl << "\n    ***************************       " << endl <<
endl;

                int n, k;
                cout << "\nEnter Number of days : ";
                cin >> n;

                float* price = new float[n];
                productPriceListGenerator(price, n, temp);

                cout << "\nPrice list initially :- " << endl;
                for(int i=0; i<n; i++)
                {
                        cout << price[i] << endl;
                }
                cout << endl;

                LCG(13, 3, 32, 7, price, n);      //library routines
                cout << endl;

                cout << "Final price list :- " << endl;
                for(int i=0; i<n; i++)
                {
                        cout << price[i] << endl;
                }
                cout << endl;

                cout << "Enter k(buys and sells one can do) : ";
                cin >> k;

                cout << "\nMaximum profit is : ";
                float profitEarned = Profit(price, n, k);
                cout << profitEarned << endl;
```

```
        update(temp, profitEarned);

        stockSpan(price, n);
        cout << endl;
        cout << "Stock span of generated final list is :- " << endl;
        for(int i=0; i<n; i++)
        {
            cout << price[i] << " ";
        }
        cout << endl;
    }
    else
    {
        char select;
        cout << "Forgot Your password?(y/n) : ";
        cin >> select;

        if(select == 'y')
        {
            forgotPassword(temp);
        }
    }

    cout << endl << "\n    ****************************       " << endl << endl;

    char select;
    cout << "\nShow your all accounts?(y/n) : ";
    cin >> select;

    if(select == 'y')
    {
        show();
    }

    cout << "Delete Account?(y/n) : ";
    cin >> select;

    if(select == 'y')
    {
        deleteAccount();
```

```
            cout << "\nRemaining Accounts :-" << endl;
            show();
        }
        cout << endl << endl;

        return 0;
}
```

OUTPUT :

```
Welcome to trading world !!

Do you want to Add an account(y/n) : y
Enter name : Shubham
Enter age : 19
User id : shubham@2k19se119
Enter password : 119

Do you want to Add an account(y/n) : y
Enter name : Taranjeet
Enter age : 20
User id : taranjeet@2k19se136
Enter password : 136

Do you want to Add an account(y/n) : n

Enter ID : shubham@2k19se119
Enter password : 119
~ Successfully Logged In ~
Hello Shubham
Your left balance is : 2500


  ****************************


Enter Number of days : 10

 1. DEBT CORPORATE BONDS(1050-1705)
 2. GOLD:Gold ETFs(7050-8028)
 3. INFRASTRUCTURE INVESTMENT TRUSTS(INVIT)(3809-4906)
 4. DABUR INDIA(5106-7000)
 5. ORIENT ELECTRIC(2104-3078)
 6. GILLETE INDIA(3400-4500)
 7. VIP INDUSTRIES(3070-3578)
 8. COLGATE(1513-3508)
 9. MARICO(370-508)
10. GODREJ CONSUMER(685-708)
11. BAJAJ CONSUMERS(182-250)

Select product : 2
Insufficient balance for this product please select product
under 2500
```

```
Select product : 1

Price list initially :-
1050
1355
1404
1517
1504
1532
1437
1121
1683
1099

Product gain/loss in different days :-
Day 1 :  69.1146
Day 2 :  -8.48741
Day 3 :  128.356
Day 4 :  -10.4844
Day 5 :  29.6206
Day 6 :  -12.4815
Day 7 :  -8.98667
Day 8 :  -14.4785
Day 9 :  148.103
Day 10 :  -0.499259

Final price list :-
1119.11
1346.51
1532.36
1506.52
1533.62
1519.52
1428.01
1106.52
1831.1
1098.5

Enter k(buys and sells one can do) : 5

Maximum profit is : 1164.93

Stock span of generated final list is :-
1 2 3 1 5 1 1 1 9 1
```

```
Stock span of generated final list is :-
1 2 3 1 5 1 1 1 9 1


  ****************************


Show your all accounts?(y/n) : y
Name : Shubham
User ID : shubham@2k19se119
Balance : 3664.93

Name : Taranjeet
User ID : taranjeet@2k19se136
Balance : 2500

Delete Account?(y/n) : y
Enter user id : taranjeet@2k19se136
Enter password : 136
account deleted

Remaining Accounts :-
Name : Shubham
User ID : shubham@2k19se119
Balance : 3664.93




[Program finished]
```

FUTURE SCOPE OF PROJECT

Our project will be able to implemented in future, as this is works as stock market simulator. And our project is also helpful in paper trading.

A stock market simulator is a program or application that attempts to reproduce or duplicate some or all the features of a live stock market on a computer so that a player may practice trading stocks without financial risk. Paper trading (sometimes also called "virtual stock trading") is a simulated trading process in which would-be investors can 'practice' investing without committing real money.

REFERENCES :

~ “Idea of stock market program for C++ computer science,” Nov. 16, 2020. [Online]. Available : http://www.leetcode.com

~ “Basic theory of stock market,” Nov. 16, 2020. [Online]. Available : https://en.m.wikipedia.org

~ www.capital.com

~ “The Data File Handling explanation for C++ computer science,” Nov. 16, 2020. [Online]. Available: https://www.youtube.com/playlist?list=PLDA2q3s0-n15f2TjU2riCPKrN9Lbnm5c0