(1)

Define a structure to represent the information of a term:-

- the cof. of the term.
- The power of the term.
- The value of the variable.
- The name of the variable.

Define a structure to represent the information of a polynomial equation:-

- The number of the terms in the equation.
- The terms themselves.

Write a program to do the followings:

- Read the information of **(N)** equations.
- Ask the user about a specific variable's name, and then do partial differentiation for all equations for the selected variable.
- Ask the user about a specific power and a specific variable's name, and then find all equations, which include the selected variable with the selected power. After that display the result of evaluation of those equations.

```
#include <iostream>
#include <stdio.h>
using namespace std;

struct Term
{
	int cof;
	int power;
	int val;
	char name;
};

struct PolyEq
{
	int num;
	Term* T;
};

void main()
{
	int N,sp,eval,f;
	char sv;
	PolyEq* P;

	cout << "Enter N" << endl;
	cin >> N;

	P = new PolyEq[N];

	for (int i = 0; i < N; i++)
	{
		cin >> P[i].num;

		P[i].T = new Term[P[i].num];

		for (int k = 0; k < P[i].num; k++)
```

```
		{
			cin >> P[i].T[k].cof;
			cin >> P[i].T[k].power;
			cin >> P[i].T[k].val;
			cin >> P[i].T[k].name;
		}

	}

	cout << "Enter specific variable name" << endl;
	cin >> sv;

	for (int i = 0; i < N; i++)
	{
		for (int k = 0; k < P[i].num; k++)
		{
			if (P[i].T[k].name == sv)
			{
				P[i].T[k].cof *= P[i].T[k].power;
				P[i].T[k].power -= 1;
			}
		}
	}

	cout << "Enter specific power" << endl;
	cin >> sp;
	cout << "Enter specific variable name" << endl;
	cin >> sv;

	eval = 0;
	for (int i = 0; i < N; i++)
	{
		eval = 0;
		for (int k = 0; k < P[i].num; k++)
		{
			if (P[i].T[k].name == sv && P[i].T[k].power == sp)
			{
				f = P[i].T[k].val;
				for (int z = 0; z <= P[i].T[k].power; z++)
				{
					P[i].T[k].val *= f;
				}

				P[i].T[k].val *= P[i].T[k].cof;
				eval += P[i].T[k].val;

			}
		}

		cout << eval << endl;
	}


}
```

(2)

Declare a structure "***Mail***" that carries the followings:

- The Name of the sender.
- The Date.
- The message's Text.
- The risk of virus attachment with the mail.
  The risk (0 : 100 %).
- The Name of the destination.

Declare another structure "***Inbox***" that carries the followings:

- Number of mails in the inbox.
- The mails themselves.

Write a program to do the followings:

- Read **(N)** inboxes from the user.
- Ask the user to enter a range of risk (say : 15 % : 35%), and then display all messages within this range in all inboxes.
- Ask the user to select a specific inbox (selection by index), and then display how many mails that has risk > 50 %, form this inbox.

```
#include <iostream>
#include <stdio.h>
using namespace std;

struct Mail
{
	char ns[100];
	int date;
	char mt[200];
	int risk;
	char nd[100];
};

struct Inbox
{
	int num;
	Mail* M;
};

void main()
{
	int N, s, e, in;
	Inbox* I;

	cout << "Enter no.of inboxes" << endl;
	cin >> N;

	I = new Inbox[N];


	cout << "Enter values" << endl;
	for (int i = 0; i < N; i++)
	{
		cin >> I[i].num;
```

```
		I[i].M = new Mail[I[i].num];

		for (int k = 0; k < I[i].num; k++)
		{
			gets_s(I[i].M[k].ns);
			cin >> I[i].M[k].date;
			gets_s(I[i].M[k].mt);
			cin >> I[i].M[k].risk;
			gets_s(I[i].M[k].nd);

		}
	}

	cout << "Enter range of risk" << endl;
	cin >> s >> e;


	for (int i = 0; i < N; i++)
	{
		for (int k = 0; k < I[i].num; k++)
		{
			if (I[i].M[k].risk >= s && I[i].M[k].risk <= e)
				cout << I[i].M[k].mt << endl;

		}
	}

	int ct = 0;
	cout << "Enter inbox" << endl;
	cin >> in;

	for (int k = 0; k < I[in].num; k++)
	{
		if (I[in].M[k].risk > 50)
		{
			ct++;
		}
	}

	cout << ct << endl;

}
```

(3) Declare a Chicken structure as the following:

- Gender of the chicken (male or female).
- Age of the chicken.
- Flu status (yes or no).

Declare another structure as the following:

- Number of chickens in the farm.
- The chickens themselves.
- City of the farm.

Write a program to do the following:

- Read information for **(N)** farms.
- Ask the user to select a city's name, and then display how many farms in this city.
- Ask the user to select a farm (by index value), and display the percentage of diseases chickens in this farm.

```
#include <iostream>
#include <stdio.h>
using namespace std;

struct Chicken
{
	char gender;
	int age;
	char IsFlu;

};

struct Farm
{
	int num;
	Chicken *c;
	char city[200];
};

void main()
{
	int N,ct=0,inf;
	Farm* f;
	char un[200];

	cout << "Enter no.of farms" << endl;
	cin >> N;

	f = new Farm[N];

	cout << "Enter information" << endl;
	for (int i = 0; i < N; i++)
	{
		cin >> f[i].num;

		f[i].c = new Chicken[f[i].num];
		for (int k = 0; k < f[i].num; k++)
		{
			cin >> f[i].c[k].gender;
			cin >> f[i].c[k].age;
```

```
                cin >> f[i].c[k].IsFlu;
        }

        gets_s(f[i].city);

    }

    cout << "Enter city's name" << endl;
    gets_s(un);
    int check;

    for (int i = 0; i < N; i++)
    {
        check = 0;
        for (int r = 0; un[r] != '\0'; r++)
        {
            if (f[i].city[r] != un[r])
            {
                check++;
            }
        }

        if (check == 0)
        {
            ct++;
        }

    }

    cout << ct << endl;
    ct = 0;
    cout << "Enter farm index" << endl;
    cin >> inf;

    for (int i = 0; i < f[inf].num; i++)
    {
        if (f[inf].c[i].IsFlu == 'y')
        {
            ct++;
        }
    }

    cout << (ct / f[inf].num)*100  << endl;

}
```

(4) Declare a structure to represent a DryCleaningProcess as the followings:

- The Tag of the clothes. *(it is a small paper tags on a shirt collar, is used to identify your clothes).*
- Type of the cloth's piece. *(shirt , coat, dress, ..etc.).*
- The price of the piece.
- The date of the process (*month:day*) only.

Declare another structure that represents a DryCleaningShop as the followings:

- The Shop's name.
- Number of processes in the shop.
- The processes themselves**.**

Write a program to do the followings:

- Read information for **(N)** shops.
- Ask the user to select a shop by index, and a date then display how many processes that did in this day.
- Ask the user to select a shop by index, and display the month that gain the maximum income.

```
#include <iostream>
#include <stdio.h>
using namespace std;

struct DryCleaningProcess
{
	char tag[200];
	char cp[200];
	int pr;
	int mon;
	int day;

};

struct DryCleaningShop
{
	char name[200];
	int num;
	DryCleaningProcess* P;

};

void main()
{
	int N, in, m, d, ct;
	DryCleaningShop* Sh;

	cout << "Number of shops" << endl;
	cin >> N;

	Sh = new DryCleaningShop[N];


	for (int i = 0; i < N; i++)
	{
		gets_s(Sh[i].name);
```

```
		cin >> Sh[i].num;
		Sh[i].P = new DryCleaningProcess[i];
		for (int k = 0; k < Sh[i].num; k++)
		{
			gets_s(Sh[i].P[k].tag);
			gets_s(Sh[i].P[k].cp);
			cin >> Sh[i].P[k].pr;
			cin >> Sh[i].P[k].mon;
			cin >> Sh[i].P[k].day;

		}

	}

	ct = 0;
	cout << "Select shop" << endl;
	cin >> in;
	cout << "Enter date" << endl;
	cout << "Month: " << endl;
	cin >> m;
	cout << "Day: " << endl;
	cin >> d;

	for (int k = 0; k < Sh[in].num; k++)
	{
		if (Sh[in].P[k].day == d && Sh[in].P[k].mon == m)
		{
			ct++;
		}
	}

	cout << ct << endl;

	int max = -99999;
	int tot = 0;
	int mpos;

	cout << "Select shop" << endl;
	cin >> in;
	ct = 1;

	for (; ; )
	{
		if (ct == 13)
		{
			break;

		}

		for (int k = 0; k < Sh[in].num; k++)
		{
			if (Sh[in].P[k].mon == ct)
			{
				tot+ = Sh[in].P[k].pr;

			}

		}

		if (tot > max)
		{
			max = tot;
			mpos = ct;
```

```
            }

            ct++;
        }

        cout << "Month: " << mpos << endl;

}
```

(5)

World Day Power-off : it is a day that most of big cities in the world are trying to power-off the lights to safe the electricity.

Declare a **Building** structure that include the following data:

- Type of the building (hotel, house, governmental ..etc.)
- Duration of the power-off (in minutes).

Declare a **City** structure that include the following data:

- City' name.
- Number of buildings in the city.
- buildings themselves.

Write a program to do the followings:

- Read the information for **(N)** cities.
- Ask the user to select a city and Display the ratio of its power off
  e.g.
  # of buildings in the city : 40
  # of power-off building in the city: 20
  So the ratio is : 0.50

- ask the user to select a city, and then display the type of the building who powered-off with the longest duration.

```
#include <iostream>
#include <stdio.h>
using namespace std;

struct Building
{
	char type[200];
	int min;

};

struct City
{
	char name[200];
	int num;
	Building* b;

};

void main()
{
	int N;
	City* c;
	char uc[200];

	cout << "Number of cities" << endl;
	cin >> N;
```

```
c = new City[N];

for (int i = 0; i < N; i++)
{
	cout << "enter name" << endl;
	gets_s(c[i].name);

	cout << "enter no.of buildings" << endl;
	cin >> c[i].num;

	c[i].b = new Building[c[i].num];
	cout << "enter details" << endl;
	for (int k = 0; k < c[i].num; k++)
	{
		cout << "type" << endl;
		gets_s(c[i].b[k].type);
		cin >> c[i].b[k].min;
	}

}

cout << "Enter city" << endl;
gets_s(uc);
int check, ratio=0,tot=0;

for (int i = 0; i < N; i++)
{
	check = 0;
	for (int r = 0; uc[r] != '\0'; r++)
	{
		if (c[i].name[r] != uc[r])
		{
			check++;
		}
	}

	if (check == 0)
	{
		for (int k = 0; k < c[i].num; k++)
		{
			tot += c[i].b[k].min;

		}

		ratio = tot / c[i].num;

	}

}

cout << "Ratio:" << ratio << endl;

cout << "Enter city" << endl;
gets_s(uc);
int max = -99999;
char maxp[200];

for (int i = 0; i < N; i++)
{
	max = -99999;
	check = 0;
	for (int r = 0; uc[r] != '\0'; r++)
	{
```

```
                    if (c[i].name[r] != uc[r])
                    {
                        check++;
                    }
                }

                if (check == 0)
                {
                    for (int k = 0; k < c[i].num; k++)
                    {
                        if (c[i].b[k].min > max)
                        {
                            max = c[i].b[k].min;
                            for (int z = 0; c[i].b[k].type[z] != '\0'; z++)
                            {
                                maxp[z] = c[i].b[k].type[z];
                            }
                        }
                    }
                }

        }
        cout << maxp << endl;

}
```