ΠΡΟΟΔΟΣ Γ'

Α.Μ.:

ΟΝΟΜΑΤΕΠΩΝΥΜΟ:

Θέμα 1 [3 μονάδες]

1. Γράψτε μια συνάρτηση κατακερματισμού με όνομα my_hash που να δέχεται ένα λεκτικό (std::string) και να επιστρέφει έναν ακέραιο αριθμό αθροίζοντας τις ASCII τιμές μόνο από τους χαρακτήρες του λεκτικού που βρίσκονται στις θέσεις με άρτιο δείκτη.
2. Ορίστε στο κύριο πρόγραμμα έναν πίνακα με 10 λεκτικά τα οποία θα εισάγει ο χρήστης. Εμφανίστε τη τιμή που θα επιστρέφει η κλήση της συνάρτησης κατακερματισμού για κάθε στοιχείο του πίνακα καθώς και τη θέση στην οποία θα αντιστοιχεί σε έναν υποτιθέμενο πίνακα κατακερματισμού 100.001 θέσεων.

```
#include <iostream>

using namespace std;

size_t my_hash(string& key){
	size_t sum=0;
	for(int i=0;i<key.size();i+=2)
		sum += key[i];
	return sum;
}

int main(){
	string data[10];
	for(int i=0;i<10;i++){
		cout << "Enter text: ";
		cin >> data[i];
		cout << my_hash(data[i]) << " " << my_hash(data[i]) % 100001 << endl;
	}
}
```

Θέμα 2 [2 μονάδες]

Δίνεται ένας πίνακας ακεραίων 4 X 4 που αναπαριστά ένα γράφημα ως πίνακας γειτνίασης. Γράψτε συνάρτηση που να δέχεται ως παράμετρο τον πίνακα, να εξετάζει αν το γράφημα είναι κατευθυνόμενο ή μη κατευθυνόμενο και να επιστρέφει στην πρώτη περίπτωση true, ενώ στη δεύτερη περίπτωση false.

```
#include <iostream>

using namespace std;

bool is_directed(int a[4][4]){
	for(int i=0;i<4;i++)
		for(int j=i;j<4;j++)
			if (a[i][j]!=a[j][i])
				return true;
	return false;
}
```

```
int main(){

	int a[4][4] =	{
						{0,5,3,7},
						{5,0,0,2},
						{3,0,0,0},
						{7,2,0,0}
					};
	if (is_directed(a))
		cout << "Directed graph" << endl;
	else
		cout << "Undirected graph" << endl;
}
```

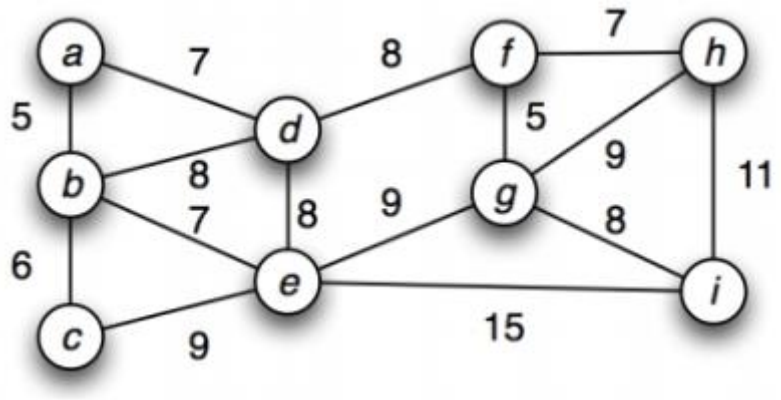
Θέμα 3 [2 μονάδες]

Γράψτε ένα πρόγραμμα που να δέχεται ακέραιες αριθμητικές τιμές από τον χρήστη. Να εμφανίζει το άθροισμα από τις διακριτές (διαφορετικές) τιμές που εισήγαγε ο χρήστης χρησιμοποιώντας είτε ένα std::set είτε ένα std_unordered_set.

```
#include <algorithm>
#include <iostream>
#include <set>

using namespace std;

int main(){
	set<int> s;
	int x;
	cout << "Enter value (-1 to stop): ";
	cin >> x;
	while (x!=-1){
		s.insert(x);
		int sum =0;
		for(int a: s)
			sum += a;
		cout << "SUM = " << sum << endl;

		// cout << "SUM = " << accumulate(s.begin(), s.end(),0) << endl;

		cout << "Enter value (-1 to stop): ";
		cin >> x;
	}
}
```

Θέμα 4 [3 μονάδες]

1. Εφαρμόστε στο ακόλουθο γράφημα τον αλγόριθμο του Dijkstra για την εύρεση των συντομότερων διαδρομών προς όλες τις κορυφές χρησιμοποιώντας ως αφετηρία την κορυφή **a**.
2. Καταγράψτε τη συντομότερη διαδρομή για κάθε κορυφή και το μήκος της.



S	a	b	c	d	e	f	g	h	i
{}	0	INF	INF	INF	INF	INF	INF	INF	INF
{a}	0	5_a	INF	7_a	INF	INF	INF	INF	INF
{a,b}	0	5_a	11_b	7_a	12_b	INF	INF	INF	INF
{a,b,d}	0	5_a	11_b	7_a	12_b	15_d	INF	INF	INF
{a,b,d,c}	0	5_a	11_b	7_a	12_b	15_d	INF	INF	INF
{a,b,d,c,e}	0	5_a	11_b	7_a	12_b	15_d	21_e	INF	27_e
{a,b,d,c,e,f}	0	5_a	11_b	7_a	12_b	15_d	20_f	22_f	27_e
{a,b,d,c,e,f,g}	0	5_a	11_b	7_a	12_b	15_d	20_f	22_f	27_e
{a,b,d,c,e,f,g,h}	0	5_a	11_b	7_a	12_b	15_d	20_f	22_f	27_e
{a,b,d,c,e,f,g,h,i}	0	5_a	11_b	7_a	12_b	15_d	20_f	22_f	27_e

Αφετηρία: a
Shortest path from vertex a to vertex a is {a} having length 0
Shortest path from vertex a to vertex b is {a b} having length 5
Shortest path from vertex a to vertex c is {a b c} having length 11
Shortest path from vertex a to vertex d is {a d} having length 7
Shortest path from vertex a to vertex e is {a b e} having length 12
Shortest path from vertex a to vertex f is {a d f} having length 15
Shortest path from vertex a to vertex g is {a d f g} having length 20
Shortest path from vertex a to vertex h is {a d f h} having length 22
Shortest path from vertex a to vertex i is {a b e i} having length 27