T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

Dersin Kodu: BIMU1055	**Dersin Adı: INTRODUCTION TO PROGRAMMING**
Dersin Öğretim Üyesi: Dr.Öğr.Üyesi Özgür Can TURNA	**Sınav Türü: Bütünleme**
Sınav Tarihi ve Süresi: 14.06.2019 (75 dk.)	**Öğrenci No:**
Öğrenci Ad-Soyad: CEVAP ANAHTARI	**Öğrenci İmzası:**

1. (30p) Write a sorting function that will order the values in the array according to their distances from the average. You can divide your solution to multiple auxiliary functions.

Example:

```
Input: arr[] = {1,2,4,5,3}
Ex Call to function:   sortAvg(arr);
Output: arr[] { 3 , 4 , 2 , 5 , 1 }
The average of the array is 3. So the distances 3:0 , 4:1 , 2:1 , 5:2, 1:2 => 0,1,1,2,2
```

Solution:

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

inline void swap(int& a, int& b) {
	int temp = a;
	a = b;
	b = temp;
}
void sortAvg(int arr[] , int size)
{
	float avg, sum =0;
	for (int i = 0; i < size; i++)
	{
		sum += arr[i];
	}
	avg = sum / size;
	for (int i = 0; i < size -1; i++)
	{
		for (int j = i+1; j < size; j++)
		{
			if (abs(arr[i] - avg) > abs(arr[j] - avg))
			{
				swap(arr[i], arr[j]);
			}
		}
	}
}

int main()
{
	int arr[] = { 1,3,5,4,2 };
	int size = sizeof(arr) / sizeof(int);
	sortAvg(arr, size);
	for (int i = 0; i < size; i++)
	{
		cout << arr[i] << " ";
	}
	return 0;
}
```

2. (30p) Write a function that find an integer in a string data. Function must return the start index of the integer value in string data.
If it is not present, the return value must be -1.

```
int findIntInStr(string data, int val);
```

Solution:

```
int findIntInStr(string data, int val)
{
	string s;
	int i, j;
	char temp;
	bool match;
	if (val < 0) { val = -val; }
	while (val > 0) { // to string
		s += (char)(val % 10 + 48);
		val = val / 10;
	}
	if (s.size() > 1) {		//take reverse.
		for (i = 0; i < s.size() / 2; i++)
		{
			temp = s[i];
			s[i] = s[s.size() - i - 1];
			s[s.size() - i - 1] = temp;
		}
	}
	for ( i = 0; i < data.size()+ 1 - s.size() ; i++)
	{  //find matching series.
		if (data[i] == s[0] )
		{
			match = true;
			for ( j = 0; j < s.size(); j++)
			{
				if (data[i + j] != s[j])
				{
					match = false;
					break;
				}
			}
			if (match) return i;
		}
	}
	return -1;
}


int main()
{
	cout << findIntInStr("Merhaba 123 x 321", 123) << endl;
	return 0;
}
```

3. (30p) Imagine that we have an integer matrix that includes numbers from 0 to 9. Write a function that finds out that if there exist any identical neighbors. If so, it will return false, otherwise true.

Ex:

True / No identical neighbors.

0	1	2	6
3	4	5	7
6	7	8	9
9	0	1	3
2	3	4	2

False / At least there are two identical neighbors.

0	1	2	6
3	4	1	7
6	7	8	9
9	0	1	3
2	3	4	2

Function prototype bool checker(int *arr , int row, int col);

Solution:

```
bool checker(int* arr, int row, int col)
{
  for (int i = 0; i < row; i++)
  {
       for (int j = 0; j < col; j++)
       {
              if ((j < col-1 && arr[i * col + j] == arr[i * col + j + 1])                          ||
                     (i< row-1 && j < col-1 &&  arr[i * col + j] == arr[(i + 1) * col + (j + 1)] ) ||
                     (i< row-1 && arr[i * col + j] == arr[(i + 1) * col + j]  )                   ||
                     (i > 0 && arr[i * col + j] == arr[(i - 1) * col + (j + 1)]))
                     return false;
       }
  }
  return true;
}
```

4. We have a class named as “Sample”. Its implementation is missing. Sample class can store any type of data in a stack fashion. According to the code implemented in main function complete, the “Sample” class implementation. You can answer all the questions in a single class definition.

```
int main()
{
   Sample s1, s2(10), s3(“Hello World”), s4 = “Another Type of assignment”;  //LINE 1
   s1 = s2;   // Copies all the elements in s2 stack to s1 stack;            //LINE 2
   s1 = s1 + s3;  // Adds the elements in s3 stack’s and s1 stack’s then store the resulting Sample in s1 LINE 3
}
```

a) (10p) Create the “Sample” class structure with necessary access modifiers etc.
b) (20p) Write necessary code blocks of Sample class needed in LINE 1.
c) (20p) Write necessary code blocks of Sample class needed in LINE 2.
d) (20p) Write necessary code blocks of Sample class needed in LINE 3.

Solution:

```
enum dataType{INTEGER,STRING,OBJECT,FLOAT};
struct dataT {
	void* val;
	dataType type;
};
class Sample
{
private:
	vector<dataT*> data;
public:
	Sample()
	{
		data.clear();
	}
	Sample(int a)
	{
		data.push_back(new dataT{ new int( a), INTEGER });
	}
	Sample(string s)
	{
		data.push_back(new dataT{ new string(s) , STRING });
	}
	Sample(const Sample& s)
	{
		this->data.clear();
		for (auto x : s.data) {
			this->data.push_back(x);
		}
	}
	void operator=(const char* p)
	{
		data.push_back(new dataT{ (void*)p , STRING });
	}

	Sample operator+(const Sample& s2)
	{
		Sample s;
		for (auto x : this->data) {
			s.data.push_back(x);
		}
		for (auto x : s2.data) {
			s.data.push_back(x);
		}
		return s;
	}
```

```
	void print()
	{
		for (size_t i = 0; i < data.size(); i++)
		{
			switch (data[i]->type)
			{
			case INTEGER:
				cout << *(int*)(data[i]->val) << endl;
				break;
			case STRING:
				cout << *(string*)(data[i]->val) << endl;
				break;
			case FLOAT:
				cout << *(float*)(data[i]->val) << endl;
				break;
			default:
				break;
			}
		}
	}
};


int main()
{
  Sample s1, s2(10), s3("Hello World"), s4 = "Another Type of assignment";  //LINE 1
  s1 = s2;   // Copies all the elements in s2 stack to s1 stack;            //LINE 2
  s1 = s1 + s3 + s4;  // Adds the elements in s3 stack's and s1 stack's then store the resulting
                      // Sample in s1 LINE 3
  s1.print();
  return 0;
}
```

You Have 160 points question try at least 100. You can solve as much as you can. Best Luck.