National University of Computer and Emerging Sciences, Lahore Campus

	Course:	**Computer Programming**	**Course Code:**	**CS103**
	Program:	**BS(Computer Science)**	**Semester:**	**Fall 2018**
	Duration:	**60 Minutes**	**Total Marks:**	**40**
	Paper Date:	**02-Oct-2018**	**Weight**	**15**
	Section:	**All**	**Page(s):**	**5**
	Exam:	**Midterm I**	**Roll No:**	

Instructions:

- Attempt all questions on **this answer booklet**. You may do your scratch work on rough sheet, but it will not be collected/marked.
- Questions during exam are not allowed. Take reasonable assumptions where needed.

Question 1 [4x5=20 Marks]: For code segments given below identify **output or error**. In case of error highlight the line that will cause the error and describe the error **in few lines**.

Note: There is no syntax error in following code segments.

Part (i)

`void main()`
 	
`{`
 	
`int* ptr[3];`
 	
`ptr[0] = new int[5];`
 	
`ptr[1] = ptr[0];`
 	
`for (int i = 0; i < 5; i++) {`
 	
`*ptr[1]=2*i;`
 	
`ptr[1]++;`
 	
`}`
 	
`ptr[1] = ptr[1] - 5;`
 	
`ptr[2] = ptr[1];`
 	
`for (int j = 0; j < 5; j++) {`
 	
`cout << *ptr[2] << " ";`
	Output/Error:
	
	0
	
	2
	
	4
	
	6
	
	8
	
	Error:
	
	Line: cout << *ptr[2] << " ";
	
	Illegal Memory Access ptr[2] is dangling pointer

```
            ptr[2]++;
      }
      cout<<endl;
      delete[] ptr[0];
      for (int j = 0; j < 5; j++)     {
            cout << *ptr[2] << " ";
            ptr[2]++;
      }
}
```

Part (ii)

```
void DoSomething(char *str1, char* str2){
      int index = 0;
      while (str2[index] != '\0')
       {
            str1[index] = str2[index];
            index++;
      }
      str1[index] = '\0';
}
int main(){
      char str1[] = "C++ Programmers
Sessional-I";
      char str2[] = "Winter is Coming";


      DoSomething(str2, str1);
      cout << str2;
       return 0;
}
```

Output/Error:

Error at line

str1[index] = str2[index];

writing out of bound

Part (iii)

```
void functionTwo(int* &p, int *q)
{
```

Output/Error:

500   1000

```
        q = new int;

        *q = *p - 100;

        *p = *q - 100;

        delete q;

}

**void functionOne(int * p, int* &q)**

{

        p = new int;

        *p = *q + 100;

        *q = *p + 100;

        functionTwo(p, q);

        delete p;

}

int main()

{

        int x = 500;

        int* ptr1=&x;

        int* ptr2 =new int;

        *ptr2 = 1000;

        cout << *ptr1 << " " << *ptr2 << endl;


        functionOne(ptr1, ptr2);

        cout << *ptr1 << " " << *ptr2 << endl;


         functionTwo(ptr1, ptr2);

        cout << *ptr1 << " " << *ptr2;

        delete ptr2;

        return 0;

}
```
	500 1200
	
	300 1200

Part (iv)

char* SomeFunction(int i, bool flag)	**Output/Error:**

```
{
	char arr[10] = "ABCDEFG";


	if(flag == true)

	{

		char* ptr = new char[strlen(arr+i)
+1];

		strcpy(ptr,arr+i);

		return ptr;

	}

	else

	{

		return arr;

	}

}

void main()

{

	char* arr2[3];

	arr2[0] = SomeFunction(3, true);

	arr2[1] = SomeFunction(1, false);

	arr2[2] = SomeFunction(0, true);


	for(int i=0; i<3; i++)

		cout<<arr2[i]<<endl;

}
```

DEFG

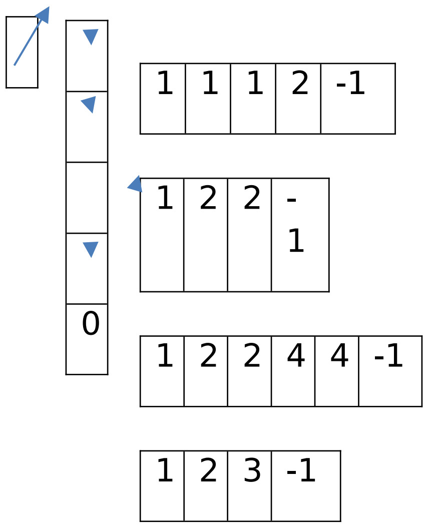
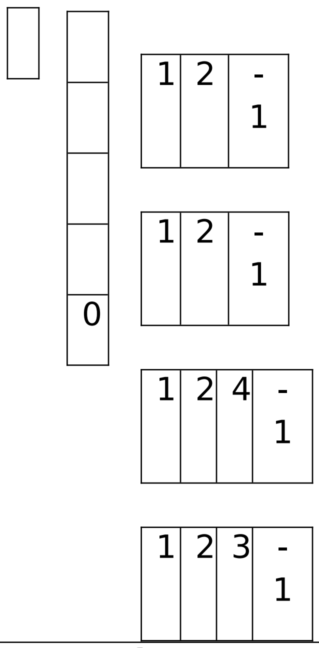
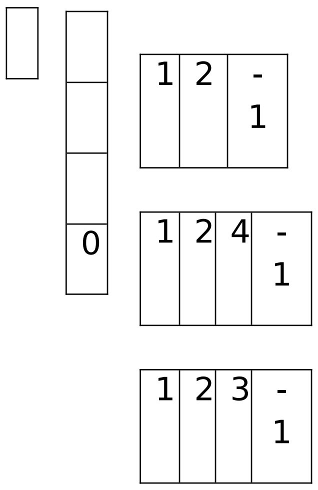
<Junk>

ABCDEFG

Error:

1- Line: return arr; Returning reference of local variable
2- Memory Leakage at the end of main.

Question 2 [20 Marks]: Write a function **void RemoveAllRepititions(int**& arr)** that takes a 2-D array “**arr”** as input and removes data repetitions in two passes. In first pass, it removes duplicate elements from each integer array. For example, integer array {1, 2, 2, 4, 4, -1} becomes {1, 2, 4, -1} after removing repetitions. In second pass, it removes repetitions if two consecutive arrays are identical. Sample run is shown below:

		
Input Array	**Array after Pass 1**	**Array after Pass 2**
	Identical consecutive elements have been removed from all rows	Identical consecutive rows removed. (First two integer arrays were identical in Pass 1)

Assume that elements in integer arrays are sorted in ascending order. Null (0) in int* array indicates end of integer arrays (delimiter in first dimension) while -1 indicates end of data in integer arrays (delimiter in second dimension). **Make sure that your program does not consume extra memory and it should not leak any memory.**

Solution: