Deadline: Friday 16th March 9.30am before the lab begins.

CFP - Pointers 1
Dynamic Memory allocations and Pointers (used as Arrays)

Estimated time 3.5-4 hours...!!!
Submission Format: A Google Document

A. Explain the difference between the following two declarations: **Estimated Time: 5mins**
int n1=n;
int& n2=n;

B. Explain the difference between the following two uses of the reference operator & and * :

int& r = n;
int* p = &n;
	int* q = p;
	n = *p;

C. When these two conditions will be false and true? x and y are two pointers **Estimated Time: 15 mins**

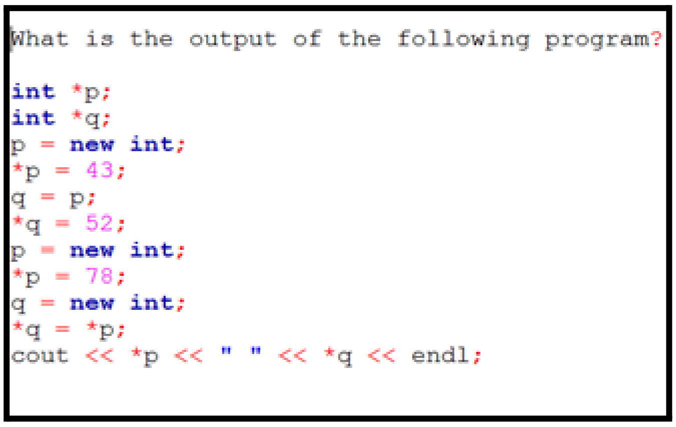
If ((x == y) & & (&x == &y))	If((x == y) & & (*x == *y))

D. What is wrong with the following statements: **Estimated Time: 30 mins**
Be careful while copy pasting the code into editor.

int& r = 22;	int* p = &44;	char c = 'w';
		char* p = c;
			char c = 'w';
			char p = &c;
				short a[32];
				for (int i = 0; i < 32; i++)
				{
				*(a) = i*i;
				a++;
				}

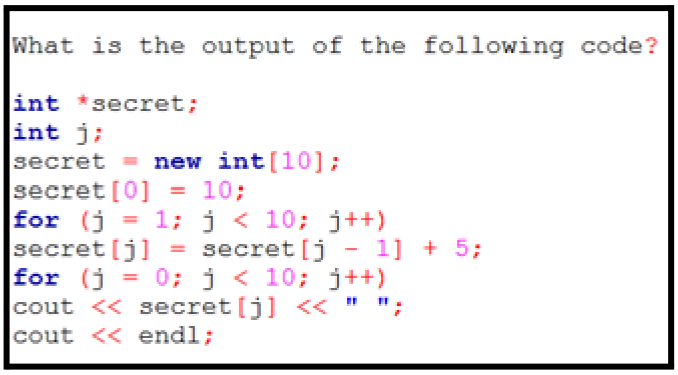
float x = 3.14159;
float* p = &x;
short d = 44;
short* q = &d;
p = q;
	int* p = new int(5);
	int* q = new int(4);
	cout << "p = " << p
	<< "q = " <<q<<
	<< ", p + q = "
	<< p + q << endl;
			Why these statements will work?
			● a[i] == *(a + i);
			● *(a + i) == i[a];
			● a[i] == i[a];	

F. Debug(STEP BY STEP EXECUTION) and see what is the output of the following programs: CLEARLY DRAW ITS PICTURE (reading the watches and every value inside every variable). **Estimated Time: 30 mins**



```
What is the output of the following program?

int *p;
int *q;
p = new int;
*p = 43;
q = p;
*q = 52;
p = new int;
*p = 78;
q = new int;
*q = *p;
cout << *p << " " << *q << endl;
```



```
What is the output of the following code?

int *secret;
int j;
secret = new int[10];
secret[0] = 10;
for (j = 1; j < 10; j++)
secret[j] = secret[j - 1] + 5;
for (j = 0; j < 10; j++)
cout << secret[j] << " ";
cout << endl;
```

E. Write a function which should take two pointers, one **int * p1** to an array's beginning and one to anywhere else after the beginning **int * p2** , and third parameter as a value **T** and search within that range **(p1 till p2)** whether the value T exists within that range if yes return the address of that placement. Clearly define its prototype. **Estimated Time: 20 mins**

F. Write a function that is passed an array of n pointers to floats and returns a newly created HEAP-array that contains those n float values. **Estimated Time: 20 mins**

Deadline: Friday 16th March 9.30am before the lab begins.

G. Write a function Swap which must swap two pointers, Test using Debugging and submit the screen-shots.
Estimated Time: 15 mins

H. **All of the following lines follow these declarations (only insert one snippet at a time, all are independent of each other):** **Estimated Time: 30 mins**

```
int a=2,b=7,c=11;
int * aptr=&a, * bptr=&b, * cptr=&c;
int x[3]={5,9,11};
char y[6]={'H','E','L','L','O','\0'}, *sptr=NULL;
```

LNo	CODE	Output
1	`cout<<&a<<" "<<&b<<" "<<&c;`	
2	`cout<<*aptr<<" "<<*bptr<<endl;`	

3	`*aptr=*bptr;`
	`cout<<a<<" "<<b;`	

4	`bptr=cptr;`
	`cout<<*bptr<<endl;`	

5	`aptr=bptr;`
	`bptr=cptr;`
	`cptr=aptr;`
	`cout<<*bptr<<" "<<*cptr<<endl;`	

6	`cout<<x<<endl;`	
7	`cout<<y<<endl;//compare with 6`	
8	`cout<<(x+2)<<endl;//compare with 6`	
9	`cout<<(y+2)<<endl;//compare with 7`	
10	`cout<<*(x+2)<<" "<<*(y+2)<<endl;`	
11	`cout<<x[2]<<" "<<y[2]<<endl;`	
12	`cout<<&x[2]<<endl;//compare with 8`	
13	`cout<<&y[2]<<endl;//compare with 9`	
14	`cout<< 1[x] <<" "<< 1[y] <<endl;//why does this work?`	
15	`cout<<*x+2<<" "<<*y+2<<endl;//compare with 10`	

16	`aptr=x;`
	`cout<<*aptr<<" "<<aptr[0]<< " "<<(aptr+1)[0]<<endl;`	

17	`aptr=x+1;`
	`cout<<*aptr<<endl;//compare with 16`	

18	`sptr=y;`
	`cout<<*sptr<<endl;`	

19	`sptr=y;`
	`sptr++;`
	`cout<<sptr<<endl;//compare with 18`	

20	`(&a)[0]=-11;`
	`cout<<a<<endl;`	

H. **Write a function which prints every address of an integer array passed in parameter. The size of the array is also passed to the function.** **Estimated Time: 10 mins**

I. Write a function that sorts an array of integers without using the subscript [ ] operator.
Estimated Time: 15 mins

Deadline: Friday 16th March 9.30am before the lab begins.

Deadline: Friday 16th March 9.30am before the lab begins.

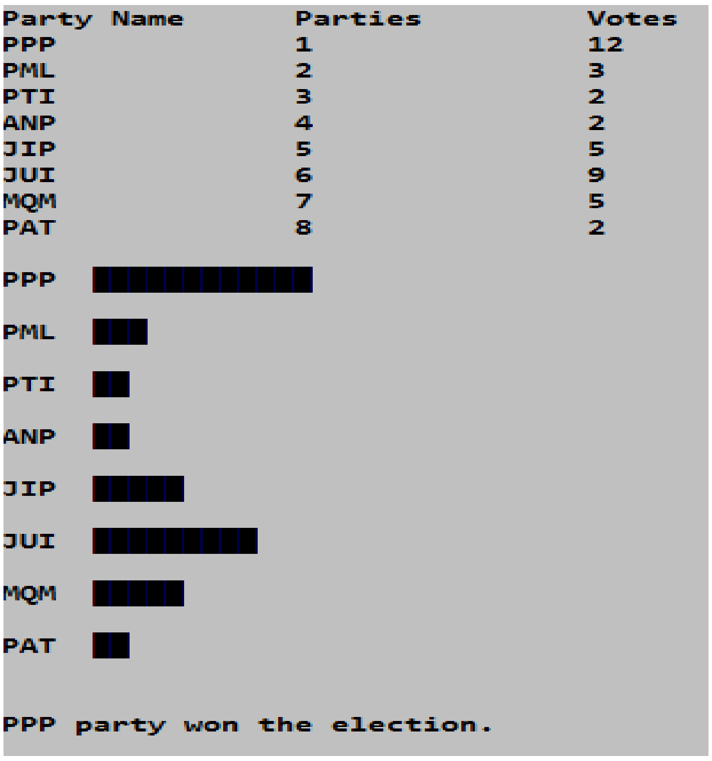
Part 2 - Dynamic Memory allocations and Pointers (used as Arrays)

For the following questions, Using [ ] operator is not allowed (other then of-course allocating the memory using new operator) to read data at a particular index of an array, instead use dereferencing (*) operator. Also you have to allocate array dynamically (using new operator: or malloc function, you may google it, and delete memory in the end using operator C++: delete or calling C's free function). PLEASE FIRST LOOK AT Q# 13 which I have solved it for you in the email.

One Solved Example of Question # 13th (Code-Solution) is here.

1. Election happened in a country with 8 parties fighting inside the election. Make a program to check who won the general election. Read the data from **Votes.txt** file.

   NOTE: **Votes** must be read inside a dynamically allocated array. The **PVotes**(Party votes count) Array can handled using two different array one for IDs and one for Frequencies(vote count casted to each Party) both dynamically allocated. .
   Expected out:



```
Party Name       Parties          Votes
PPP              1                12
PML              2                3
PTI              3                2
ANP              4                2
JIP              5                5
JUI              6                9
MQM              7                5
PAT              8                2

PPP   ████████████

PML   ███

PTI   ██

ANP   ██

JIP   █████

JUI   █████████

MQM   █████

PAT   ██


PPP party won the election.
```

2. **Bubble-Sort/Selection-Sort/Count-Sort(where values are from 1 to 100).** Data must be read in a dynamically allocated array (where pointer is in main but in the ReadData function it will be passed by reference and dynamically allocated array's location must be tracked by the referenced pointer passed to the function) and sorted in three possible ways.
3. Write a program which reads the data from file (first it must have the data size and then followed by the data) to print all unique elements in an array in sorted order.
4. Write a program to merge two sorted arrays of same size sorted in descending order. The data has to be read from the file. Make sure the new array is dynamically allocated with the size of both arrays. **NOTE: NO EXTRA MEMORY SHOULD BE ALLOCATED.**
5. Read the data from file **Data.txt**. And Compute **Mean/Mode/Median** of the data.
6. Write a program in which to count the frequency of each element of an array. The data must be read from the file and the format of the file is that the data ends with -1. Hence the array should be dynamically growable array **(allocated through pointer and with every new addition it should be grown again).**

   The Way to ReGrow(Code-Solution) is shown in this example: Follow this link

7. Write a program to find the maximum and second maximum and minimum and second minimum element in an array (read from a file and loaded into an array allocated dynamically on heap using new operator).
8. Write a program to separate odd and even integers in separate arrays. Properly design functions what parameters should be passed? **MAKE SURE YOU HAVEN'T wasted a single space of any data i.e. Exactly allocated Even Array and Odd array after counting how many size it should be?**

Deadline: Friday 16th March 9.30am before the lab begins.

9. Write a program to sort elements of the array in this order. The even indexed elements should be in descending order and odd indexed elements must be in ascending order. Data must be read inside a dynamically allocated array.
10. Write a program insert New value in the array (sorted list).
11. Write a program to delete an element at desired position from an array.
12. Read the Sorted array and do Binary Searching.
13. Read the array as input (where array is allocated dynamically exactly how much is needed), and output whether the read array is Sorted array or not.