CMPE 312- OPERATING SYSTEMS

EXERCISE 3A

In this week's lab session, we are going to learn about dynamic allocation of the memory and use of the pointers in C programming language.

The main essential difference between Java and C programming languages is that Java has a system called garbage collector which is the main underlying mechanism to operate the memory allocation and object/ variable creation dynamically. In C programming language, these kind of actions and decisions can be performed by manually with the leading of the programmer himself/ herself.

The programmers are able to create a variable(array, function, struct etc.) which has a designated and known memory address on computer memory, also we can determine when to create and when to annihilate these dynamically created objects by us.

The main advantage of this kind of management of memory that once we got into tackling with machine level system details or we might wish to make a decision that will be critical about performance, we could take advantage and make us of pointers and dynamically created objects.

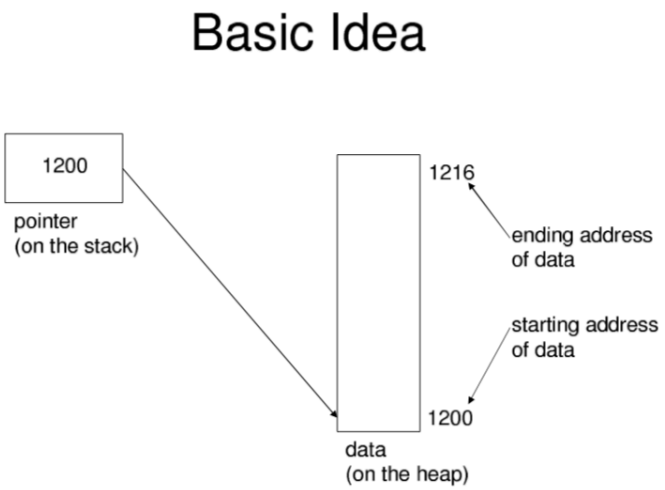
Pointers:

There are two types of memory areas on the computers' memory (by saying memory, we refer RAM of our computers), which are Stack and Heap memory areas.

The Stack is the place where all local variables are stored. For instance; when you declare an integer variable named as x in a function as local variables, it is stored in stack and as soon as function ends, the variable name and its space in memory is gone.

The Heap is an area of memory that the user handles explicitly. User requests and releases the memory through system calls. If a user forgets to release memory, it does not get destroyed. User maintains a handle o memory allocated in the heap with a pointer.

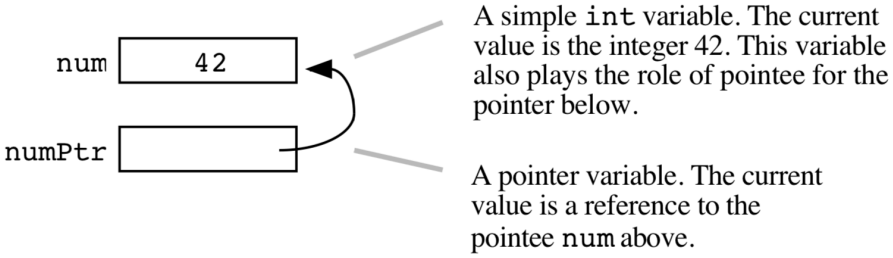
A pointer is simply a local variable that refers to a memory location on the heap. Accessing the pointer, actually references the memory on the heap.



Example:

```c
#include <stdio.h>
void foo(){

}
int main(void) {
  int num=42;//An integer value
  int *numPtr;// A pointer variable
  numPtr=&num;//Assigning the value of int to numPtr pointer.
  printf("%p\n",numPtr);//Memory address of the pointer, which is now the num's
  address.
  printf("%d\n",*numPtr);//The value that the pointer is pointing.
  return 0;
}
```



Built-in Function to allocate/de-allocate the Memory:

malloc(): Allocates requested size of bytes and returns a pointer first byte of allocated space

free(): deallocate the previously allocated space

Example:

```c
#include <stdio.h>
#include<stdlib.h>

int main() {
  int *ptr, i,size;
  printf("Enter number of elements\n");
  scanf("%d",&size);

  ptr = malloc(size*sizeof(int));//Allocate the memory with the number of ints that the user determines.
  for(i=0;i<size;i++){
    scanf("%d",(ptr+i));//To fill in the current memory space
    printf("%d\n",*(ptr+i));//To display the given value
  }

  free(ptr);// To de-allocate the memory

}
```

Task: Display the sum of the all entered numbers by the user.