Lab 10 is on pointers. It covers: addresses, pointers, the heap, dynamic memory allocation (new, delete, delete [])
It also has a very brief introduction to linked lists, a basic data structure.

With assistance from the provided startup code, create a program that uses memory as shown in the memory diagram below. Wherever possible, set the values of all variables indirectly using pointers. Use pointer dereferencing to set the values of the "pointed to" variables. You can "dereference" pointers with the '*' operator. For example, to set **price** to $19.95, do NOT use the variable price. Instead, use the pointer variable *p_price. To set pi_div_2, dereference p_PI to get 3.141592, divide it by two, and place the result into PI_div_2.

To create the "linked list" containing Person "Harry" and Person "Sally", you need the following data structure:

```
class Person {
  string name;      // name of the person
  Person *next;    // pointer to the next person in the list
};
```

To create instances of Person, you will use:

1) static allocation: Person wizard will be on the stack;
2) dynamic allocation: Persons "Harry" and "Sally" are allocated on the heap

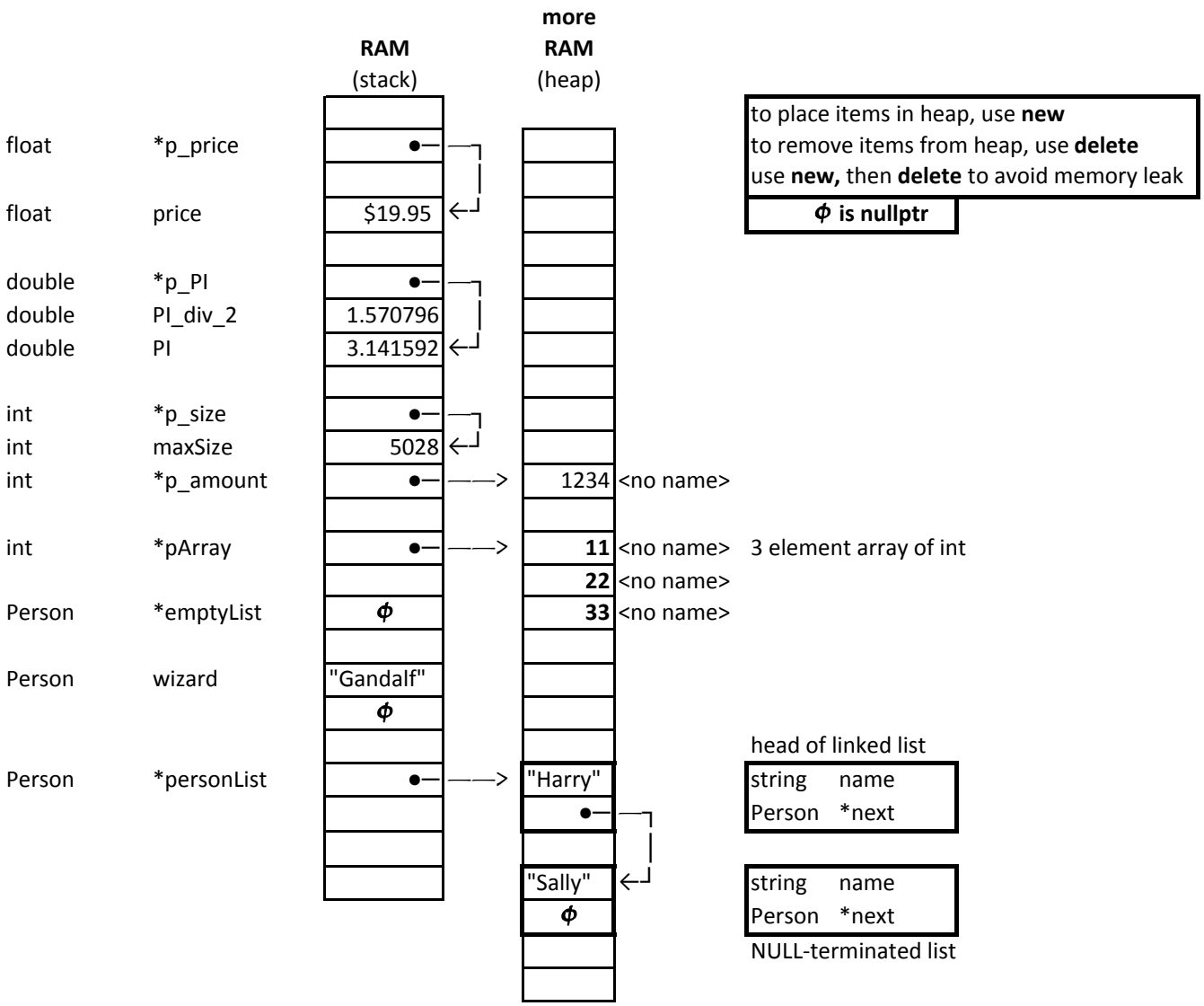
To manipulate an object of class Person, you need to create some constructors, getters and setters to fill in the name and set the pointers, or, you could make the data members public and use "dot" notation, such as: Person wizard; wizard.name="Gandalf"; wizard.next=nullptr; For dynamically allocated instances, you can use arrow notation, such as: personList -> name="Harry".

The keyword **nullptr** is the C++11/14 recommended way of setting a pointer to NULL or 0.

To create variables on the heap, with no names, you have to use new to dynamically allocate the memory. See sample output (bottom page). **IMPORTANT: Every new should be matched by a delete to prevent a memory leak! IMPORTANT: Make sure you delete all memory you dynamically allocate before leaving main(). Don't allow any memory leaks. Match all new with delete.** When deleting an array, remember to use delete [] array.

After items are placed in memory, use cout statements to output the addresses and values of all variables. Output the addresses of all variables (both pointer and named) in hexadecimal format, and output the contents (values) of all int, float, double and string variables.

You cannot force the memory layout to be exactly in the order as shown below. As a programmer, you don't control exactly WHERE in memory the compiler places variables. The important thing is for you to create the pointer variables, value variables, and set them up (nearly) as shown below.


RAM
(stack)
more
RAM
(heap)
to place items in heap, use new
to remove items from heap, use delete
use new, then delete to avoid memory leak
ϕ is nullptr
float *p_price
float price
$19.95
double *p_PI
double PI_div_2
1.570796
double PI
3.141592
int *p_size
int maxSize
5028
int *p_amount
1234 <no name>
int *pArray
11 <no name>
3 element array of int
22 <no name>
Person *emptyList
33 <no name>
Person wizard
"Gandalf"
head of linked list
Person *personList
"Harry"
string name
Person *next
"Sally"
string name
Person *next
NULL-terminated list

Expected output from Lab10 on pointers: NOTE: Your addresses may be different!
Output from Lab10 memory diagram on pointers:

```
address of price    is: &0x22fe18  contents of price    is: 19.95
address of p_price  is: &0x22fe10  contents of p_price  is: 0x22fe18
The contents of *p_price is:    19.95

address of PI       is: &0x22fe08  contents of PI       is: 3.14159
address of PI_div_2 is: &0x22fdf8  contents of PI_div_2 is: 1.5708
address of p_PI     is: &0x22fe00  contents of p_PI     is: 0x22fe08
The contents of *p_PI is:    3.14159

address of maxSize  is: &0x22fdf4  contents of maxSize  is: 5028
address of p_size   is: &0x22fde8  contents of p_size   is: 0x22fdf4
The contents of *p_size is: 5028

address of p_amount is: &0x22fde0  contents of p_amount is: 0x355bf0
The contents of *p_amount is: 1234
After delete, the contents of p_amount is: 0x355bf0
After reset to nullptr, the contents of p_amount is: 0

address of pArray    is: &0x22fdd8  contents of pArray    is: 0x355bf0
address of pArray[0] is: &0x355bf0  contents of pArray[0] is: 11
address of pArray[1] is: &0x355bf4  contents of pArray[1] is: 22
address of pArray[2] is: &0x355bf8  contents of pArray[2] is: 33
After delete [], the contents of pArray is: 0x355bf0
After reset to nullptr, the contents of pArray is: 0

address of emptyList is: &0x22fdd0  contents of emptyList is: 0

static (uses: Person wizard):
address of wizard      is: &0x22fdc0
address of wizard.name is: &0x22fdc0  contents of wizard.name is: Gandalf
address of wizard.next is: &0x22fdc8  contents of wizard.next is: 0

dynamic (uses: personList, Person("Harry"), Person("Sally"):
address of personList       is: &0x22fdb8  contents of personList       is: 0x355c20
address of personList->name is: &0x355c20  contents of personList->name is: Harry
address of personList->next is: &0x355c28  contents of personList->next is: 0x3565f0
  <follow link to next Person on personList>
address of personList->next->name is: &0x3565f0  contents of personList->next->name is: Sally
address of personList->next->next is: &0x3565f8  contents of personList->next->next is: 0
```