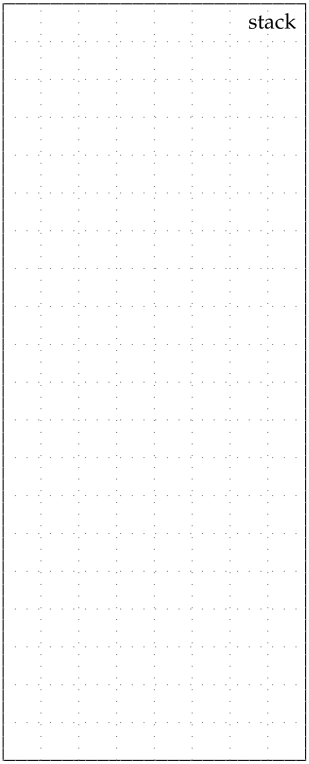
```c
#include <stddef.h>

typedef struct _node_t
{
  int             value;
  struct _node_t* next_p;
}
node_t;

int main( void )
{
  node_t node0;
  node0.value  = 3;
  node0.next_p = NULL;

  node_t node1;
  node1.value  = 4;
  node1.next_p = &node0;

  node_t node2;
  node2.value  = 5;
  node2.next_p = &node1;

  int sum = 0;
  node_t* curr_p = &node2;
  while ( curr_p != NULL ) {
    sum += curr_p->value;
    curr_p = curr_p->next_p;
  }
  return 0;
}
```



zyBooks The course zyBook includes a coding lab to implement a function to find the maximum value in a chain of nodes.

Draw a state diagram corresponding to the execution of this program

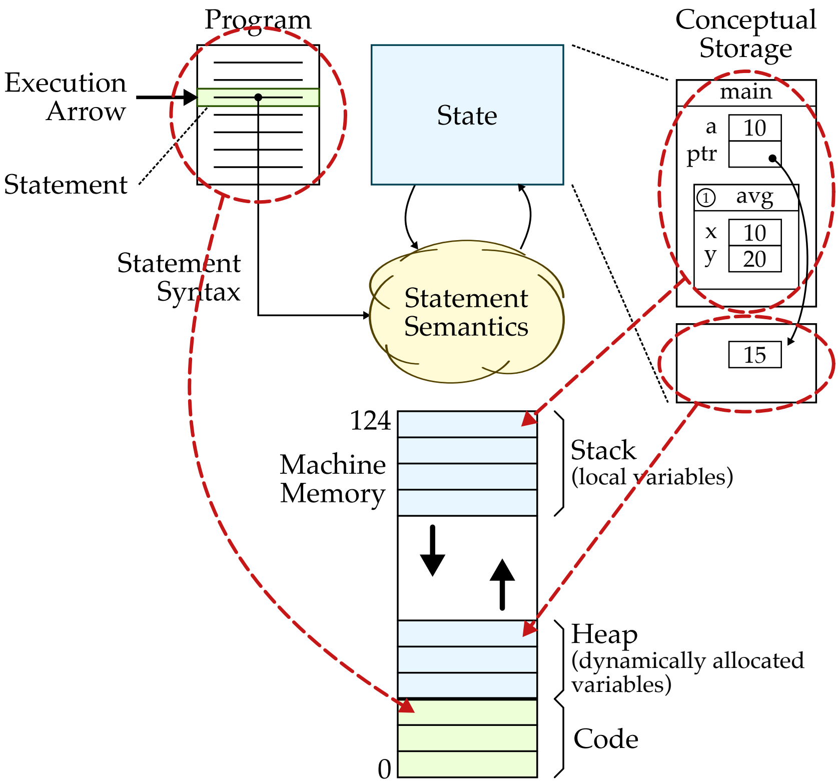
```c
int strlen( char* str )
{
  int i = 0;
  while ( str[i] != '\0' )
    i++;
  return i;
}

int main( void )
{
  char a[] = "ece2400";
  int b = strlen( a );
  return 0;
}
```

stack

zyBooks The course zyBook includes a coding lab to implement a function to copy a string from a source array to a destination array.

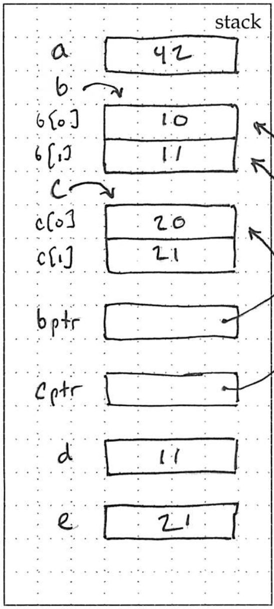
3. Mapping Conceptual Storage to Machine Memory

- Recall that our current use of state diagrams is conceptual
- Real machine uses memory to store variables
- Real machine does not use "arrows", uses memory addresses
- Heap is stored above code and grows *up*

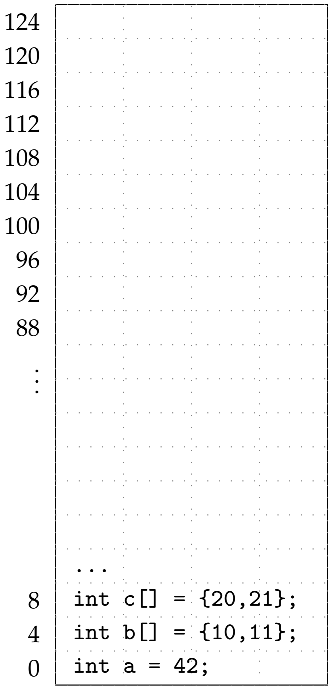


Draw both a conceptual storage and machine memory state diagram corresponding to the execution of this program

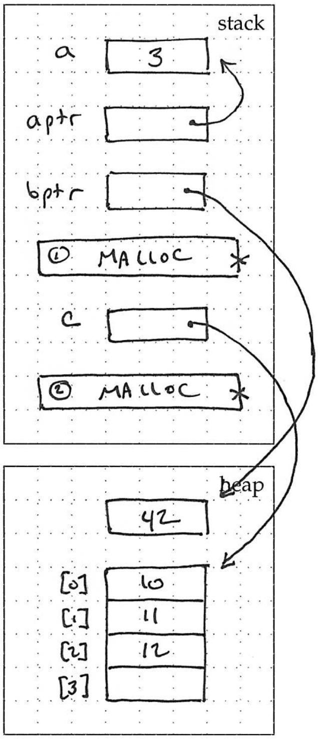
```
int a = 42;

int b[] = { 10, 11 };
int c[] = { 20, 21 };

int* b_ptr = b;
int* c_ptr = c;

b_ptr = b_ptr + 1;

int d = *b_ptr;
int e = c_ptr[1];
```



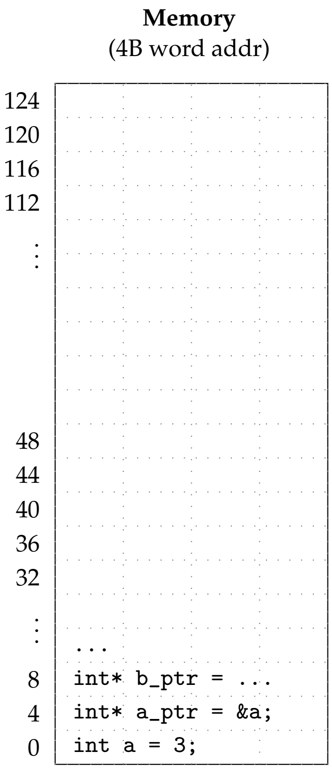
Memory
(4B word addr)



```
int   a = 3;
int*  a_ptr = &a;

int*  b_ptr = malloc( sizeof(int) );
*b_ptr = 42;

int*  c = malloc( 4 * sizeof(int) );
c[0] = 10;
c[1] = 11;
c[2] = 12;
```



stack
a
3
aptr
bptr
① MALLOC
c
② MALLOC
heap
42
[0]
10
[1]
11
[2]
12
[3]



Memory
(4B word addr)
124
120
116
112
⋮
48
44
40
36
32
⋮
8
4
0
...
int* b_ptr = ...
int* a_ptr = &a;
int a = 3;