

Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: Pointers and Dynamic Memory – Part 2

Quick Recap of Relevant Topics



- Variables and pointers to variables in C++
- “Address of” operator: &
- “Content of” operator: *
- Storage on stack segment and data segment (heap)
- “new” construct in C++

Overview of This Lecture



- Persistence of dynamically allocated memory
- Explicitly de-allocating dynamically allocated memory
- Good programming practices when de-allocating dynamic memory

Dynamically Allocating Memory in a Function

```
int * readQuizMarks(int n);  
int main()  
{ int numStudents; int * qMarks;  
  cout << "Given student count: " << endl;  
  cin >> numStudents;  
  cout << "Give marks of students" << endl;  
  qMarks = readQuizMarks(numStudents);  
  // Print max and min marks  
  return 0;  
}
```

Dynamically Allocating Memory in a Function

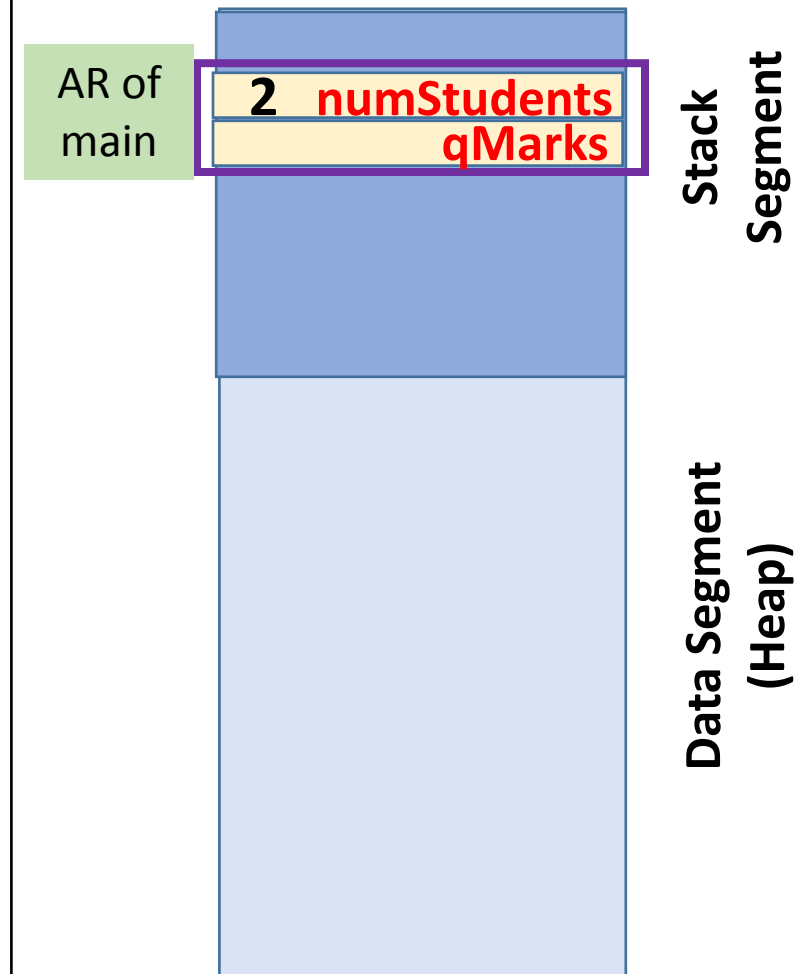


```
int * readQuizMarks(int n);  
int main()  
{ int numStudents; int * qMarks;  
  cout << "Given student count: " << endl;  
  cin >> numStudents;  
  cout << "Give marks of students" << endl;  
  qMarks = readQuizMarks(numStudents);  
  // Print max and min marks  
  return 0;  
}
```

```
// PRECONDITION: n > 0  
int * readQuizMarks(int n)  
{ int * marks, i;  
  marks = new int[n];  
  if (marks == NULL) {return NULL;}  
  for (i = 0; i<n; i++) {cin >> marks[i];}  
  return marks;  
}  
// POSTCONDITION: Marks stored  
// in dynamic array that is returned
```

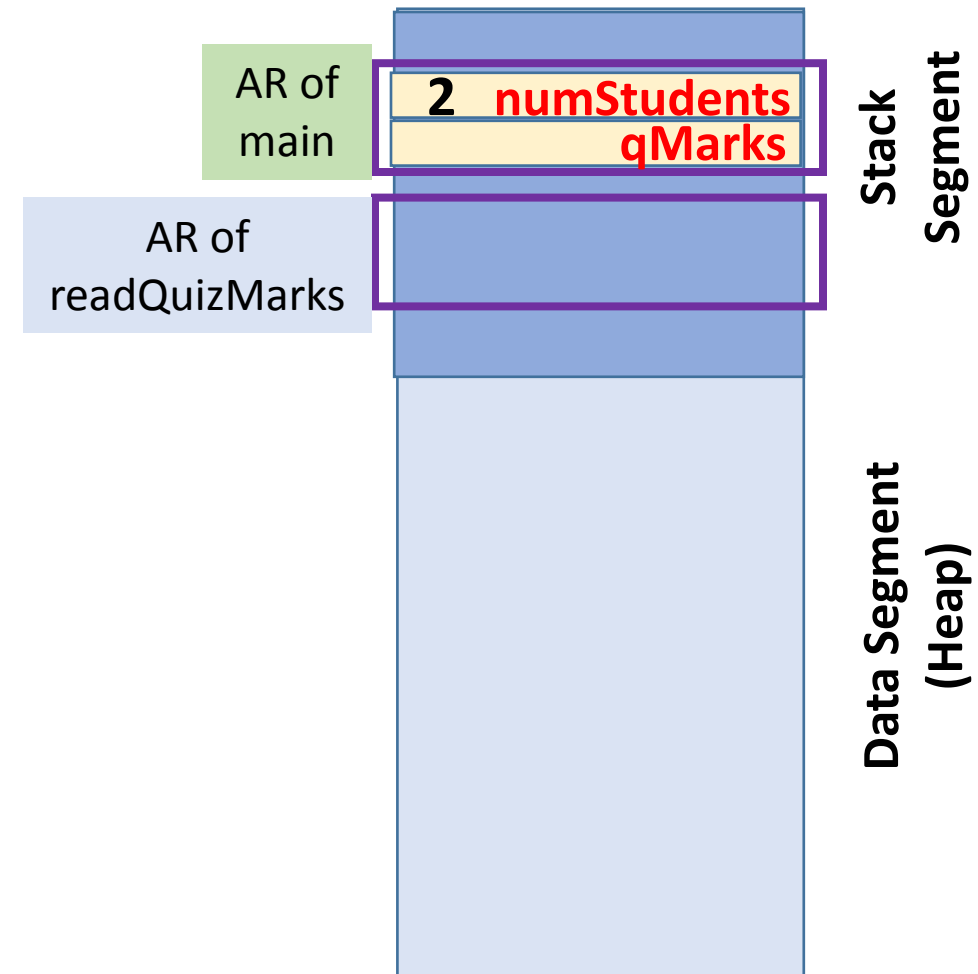
Dynamically Allocating Memory in a Function

```
int main()
{ int numStudents; int * qMarks;
  cout << "Given student count: " << endl;
  cin >> numStudents;
  cout << "Give marks of students" << endl;
  qMarks = readQuizMarks(numStudents);
  // Print max and min marks
  return 0;
}
```



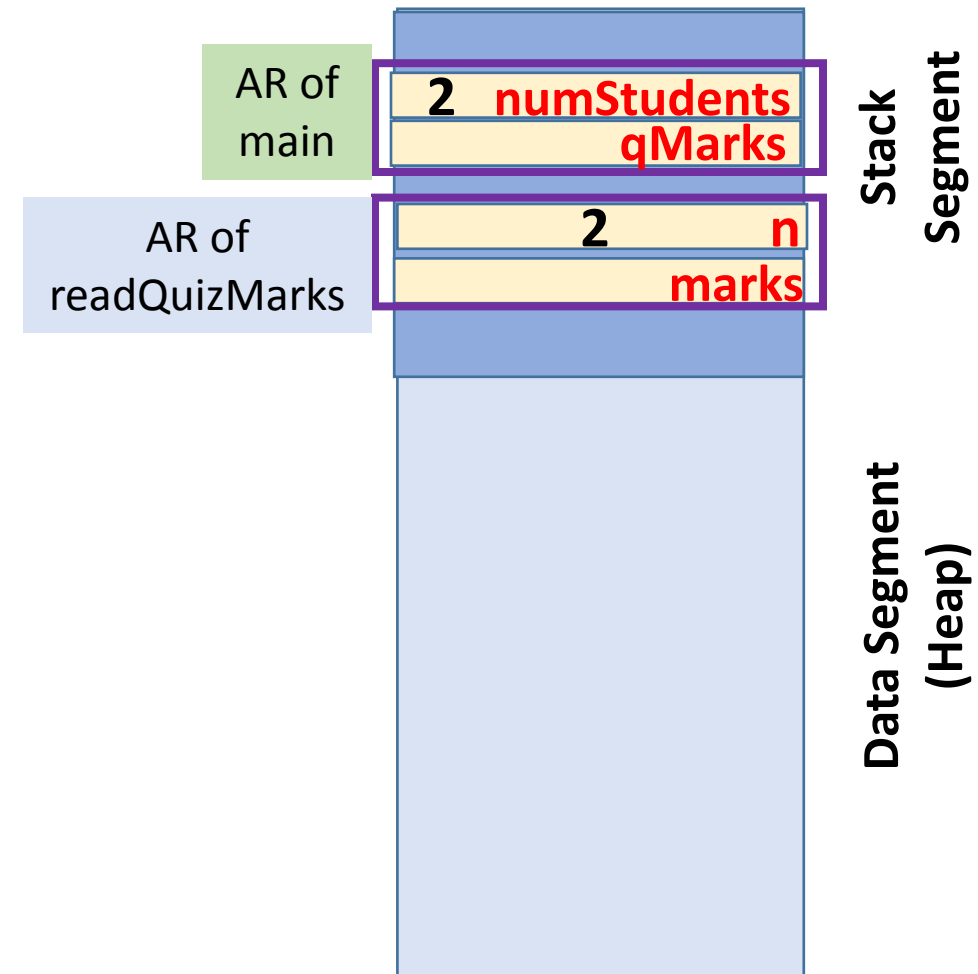
Dynamically Allocating Memory in a Function

```
int * readQuizMarks(int n)
{ int * marks, i;
  marks = new int[n];
  if (marks == NULL) {
    return NULL;
  }
  for (i = 0; i < n; i++) {
    cin >> marks[i];
  }
  return marks;
}
```



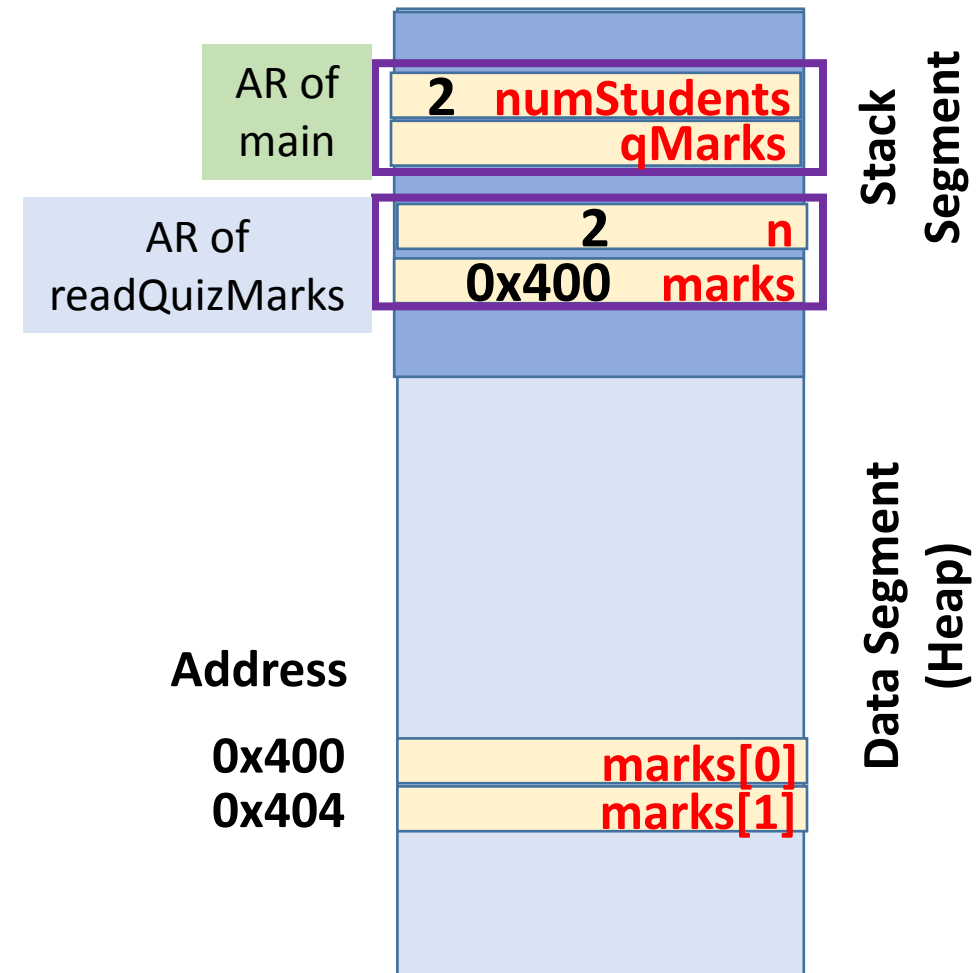
Dynamically Allocating Memory in a Function

```
int * readQuizMarks(int n)
{ int * marks, i;
  marks = new int[n];
  if (marks == NULL) {
    return NULL;
  }
  for (i = 0; i < n; i++) {
    cin >> marks[i];
  }
  return marks;
}
```



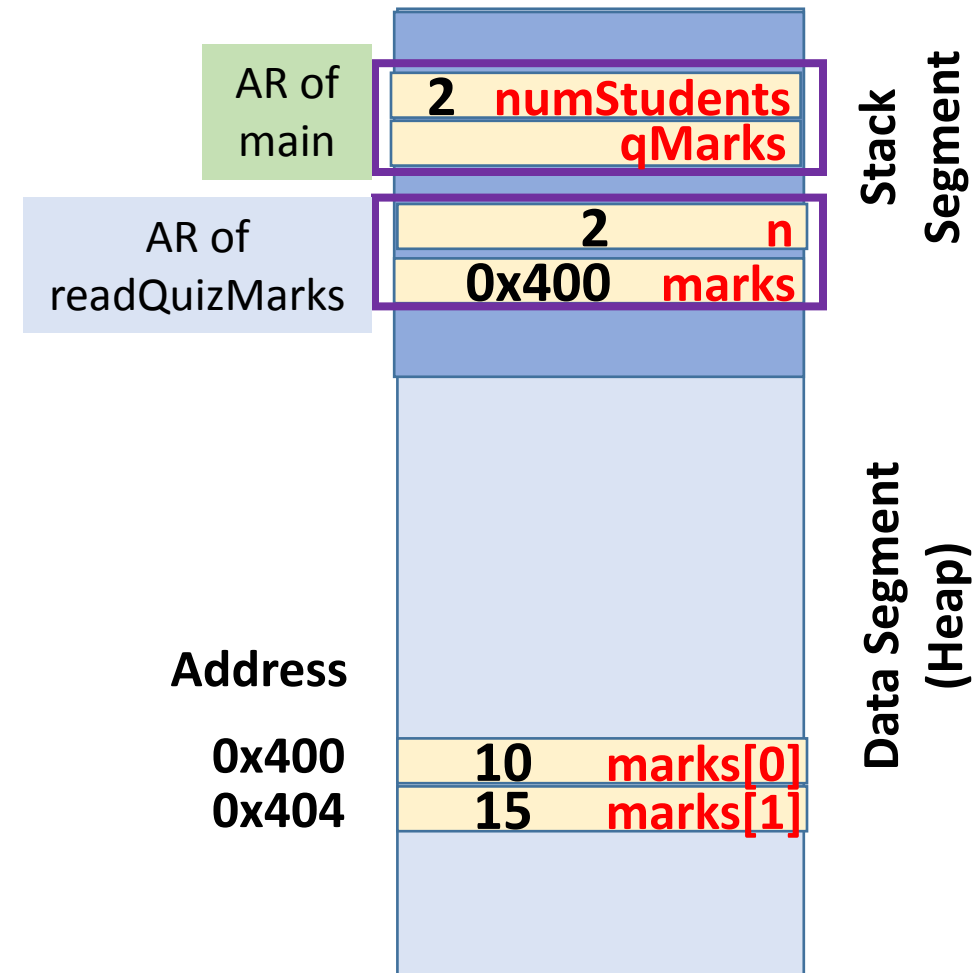
Dynamically Allocating Memory in a Function

```
int * readQuizMarks(int n)
{ int * marks, i;
  marks = new int[n];
  if (marks == NULL) {
    return NULL;
  }
  for (i = 0; i < n; i++) {
    cin >> marks[i];
  }
  return marks;
}
```



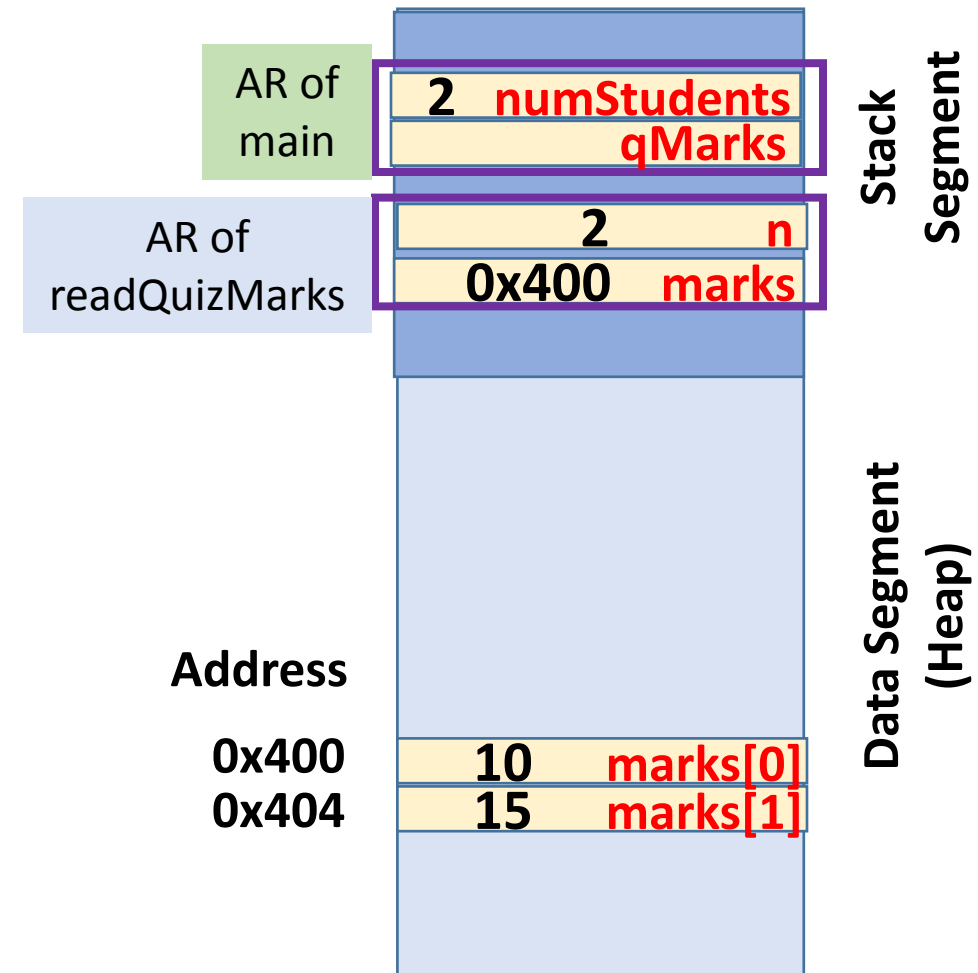
Dynamically Allocating Memory in a Function

```
int * readQuizMarks(int n)
{ int * marks, i;
  marks = new int[n];
  if (marks == NULL) {
    return NULL;
  }
  for (i = 0; i < n; i++) {
    cin >> marks[i];
  }
  return marks;
}
```



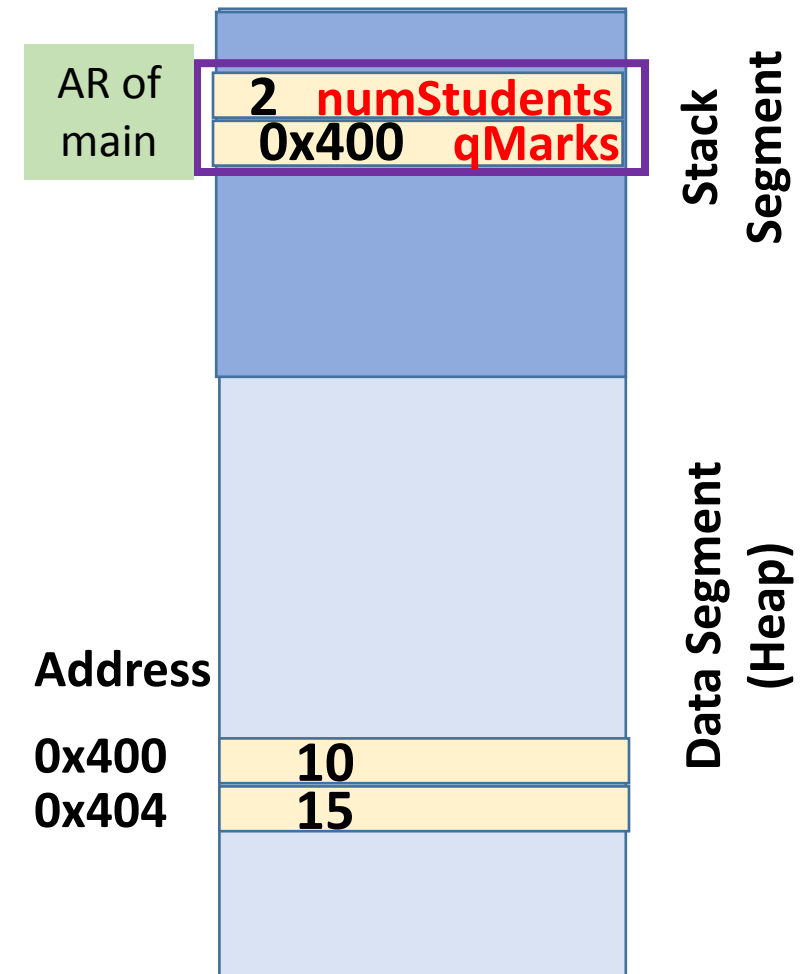
Dynamically Allocating Memory in a Function

```
int * readQuizMarks(int n)
{ int * marks, i;
  marks = new int[n];
  if (marks == NULL) {
    return NULL;
  }
  for (i = 0; i < n; i++) {
    cin >> marks[i];
  }
  return marks;
}
```



Can Caller Access Memory Allocated by Callee?

```
int main()
{ int numStudents; int *qMarks;
  cout << "Given student count: " << endl;
  cin >> numStudents;
  cout << "Give marks of students" << endl;
  qMarks = readQuizMarks(numStudents);
  // Print max and min marks
  return 0;
}
```



Can Caller Access Memory Allocated by Callee?

```
int main()
```

**Activation record of
readQuizMarks non-existent in
stack segment**

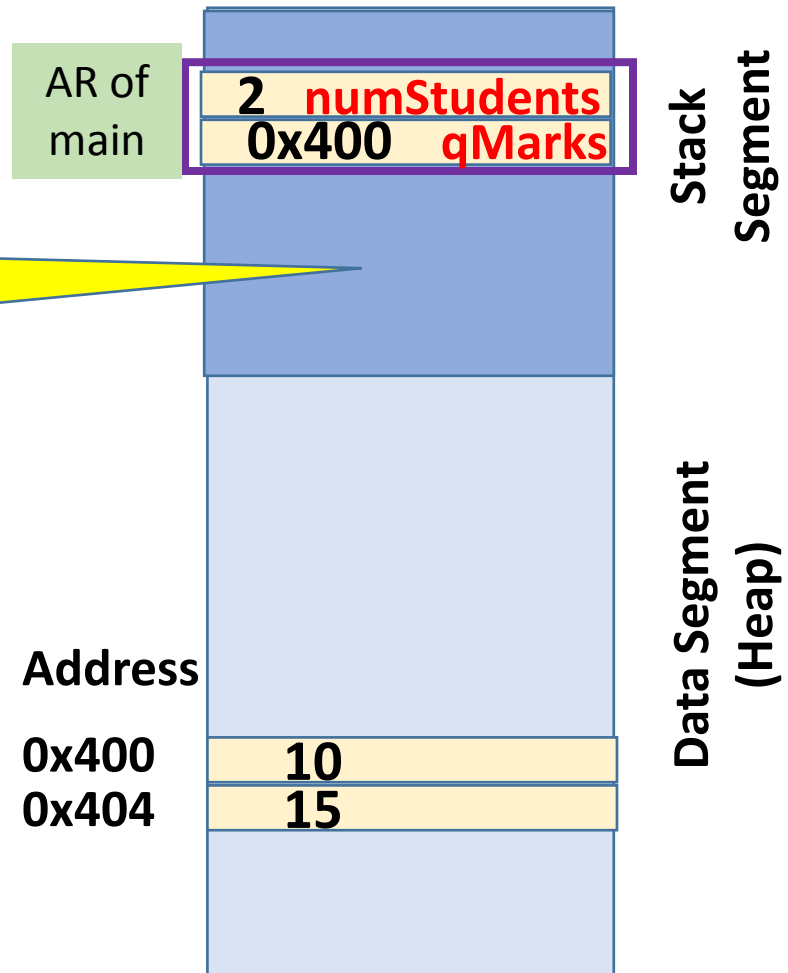
```
cout << "Give marks of students" << endl;
```

```
qMarks = readQuizMarks(numStudents);
```

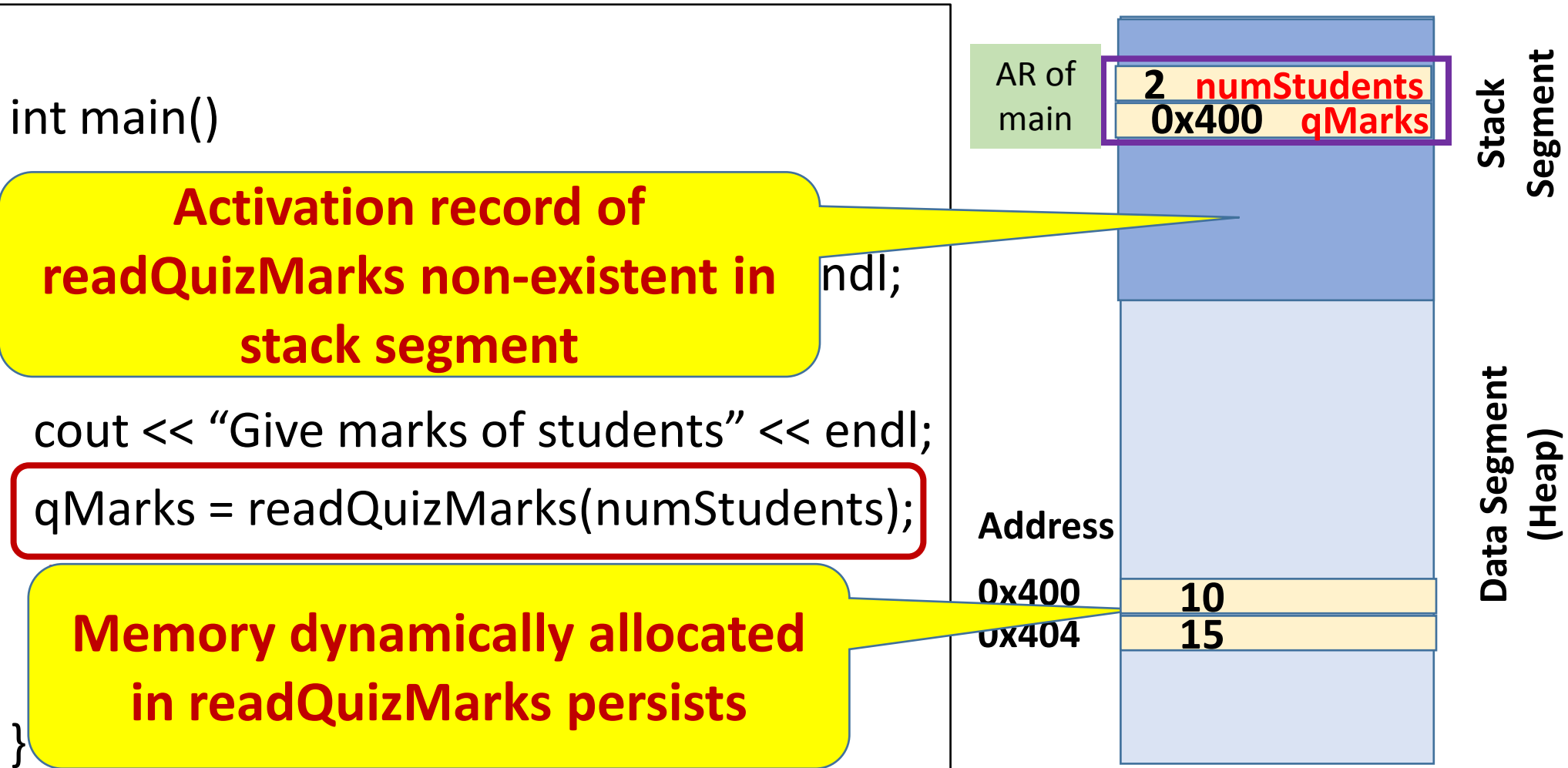
```
// Print max and min marks
```

```
return 0;
```

```
}
```



Can Caller Access Memory Allocated by Callee?



Can Caller Access Memory Allocated by Callee?

```
int main()
```

“main” has address of memory dynamically allocated in readQuizMarks.

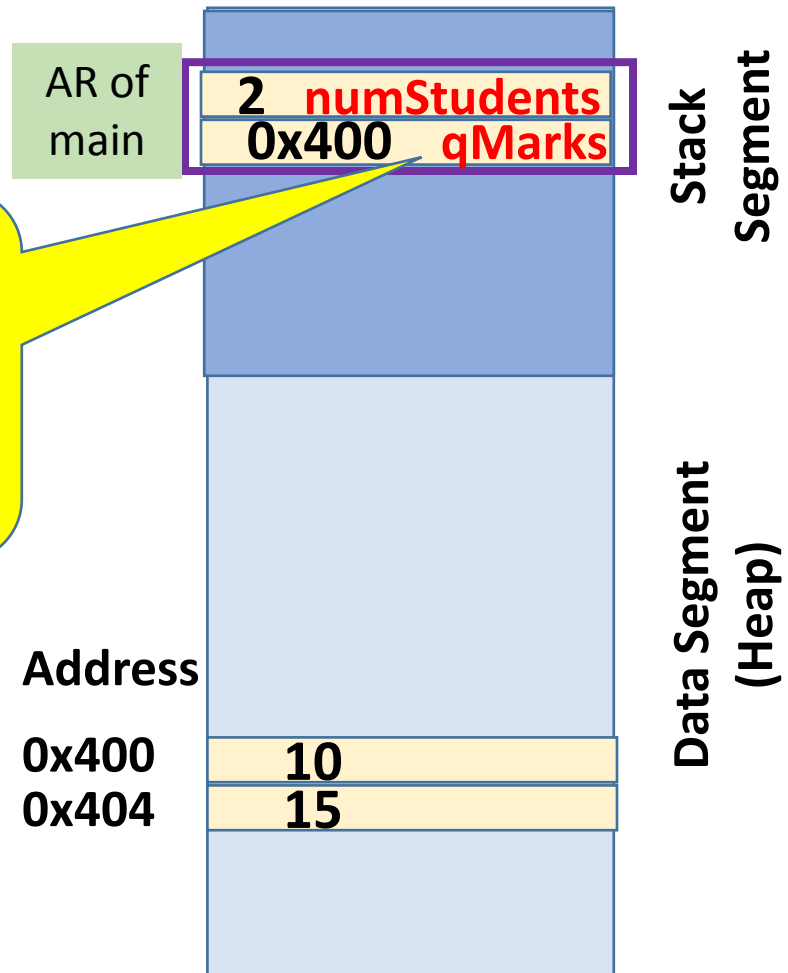
Can access this memory from “main”

```
    qMarks = readQuizMarks(numStudents);
```

```
    // Print max and min marks
```

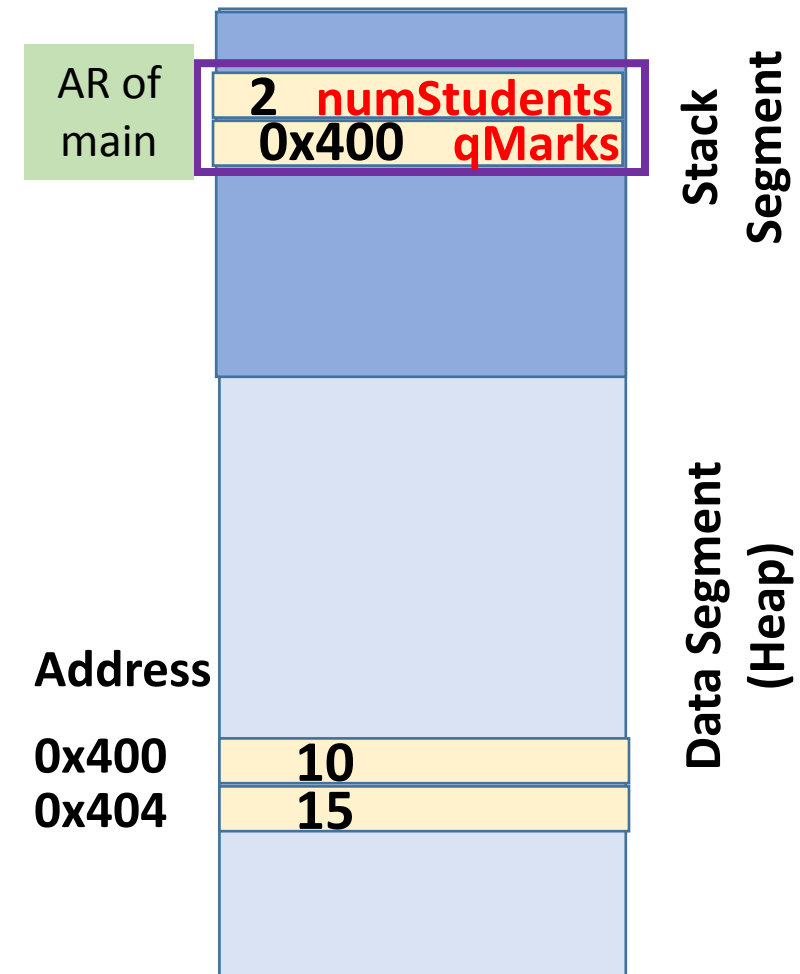
```
    return 0;
```

```
}
```



Accessing Memory Allocated by Callee

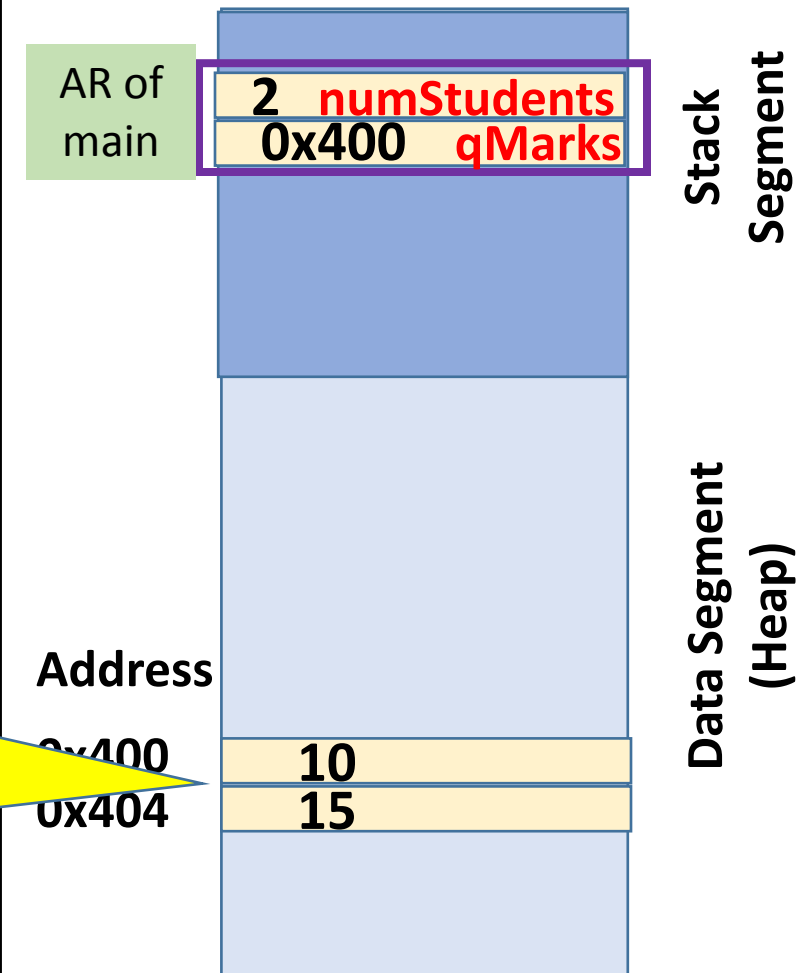
```
int main() { ... ..  
    if (qMarks == NULL) { return -1; }  
    int i, min = qMarks[0], max = qMarks[0];  
    for (i = 1; i < numStudents; i++) {  
        if (qMarks[i] < min) { min = qMarks[i];}  
        if (qMarks[i] > max) {max = qMarks[i];}  
    }  
    cout << "Min: " << min << endl;  
    cout << " Max: " << max << endl;  
    return 0;}
```



Accessing Memory Allocated by Callee

```
int main() { ... ..  
    if (qMarks == NULL) { return -1; }  
    int i, min = qMarks[0], max = qMarks[0];  
    for (i = 1; i < numStudents; i++) {  
        if (qMarks[i] < min) { min = qMarks[i];}  
        if (qMarks[i] > max) { max = qMarks[i];}  
    }  
    return 0;}
```

**When is this memory
freed (de-allocated) ?**



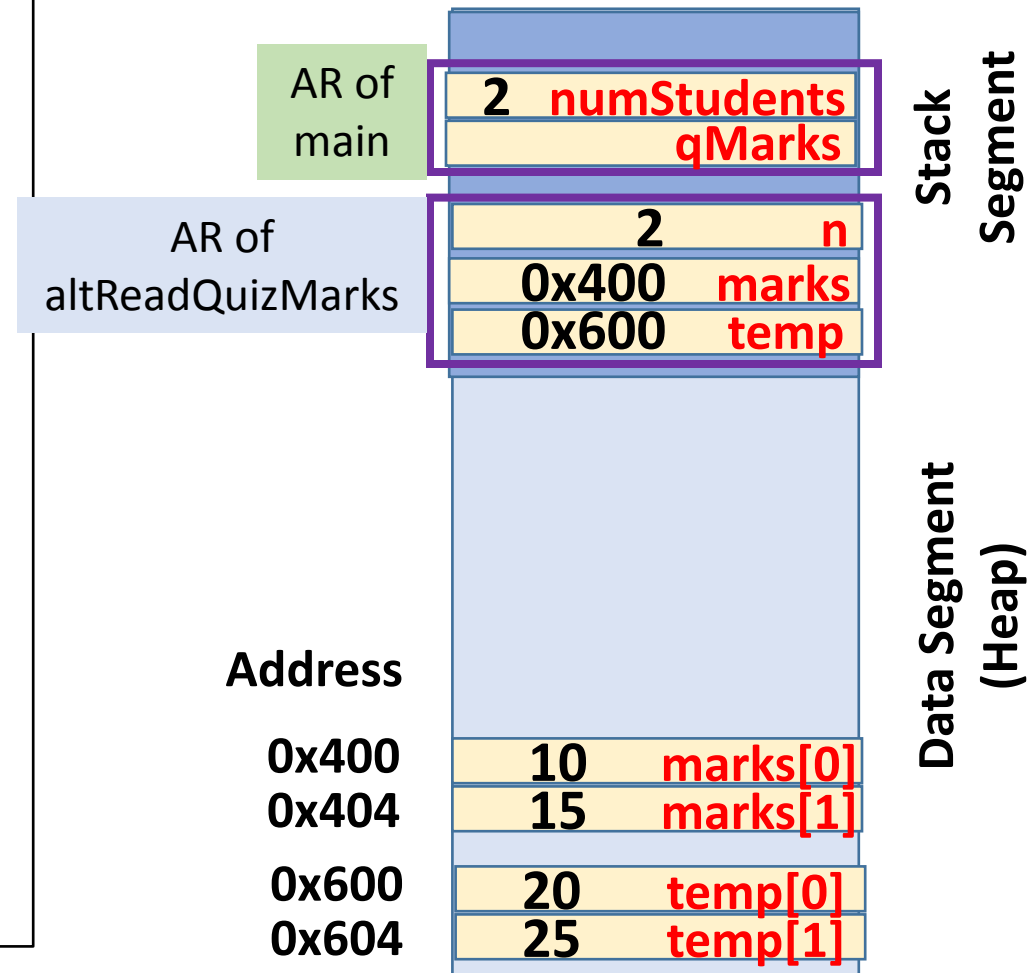
Freeing Dynamically Allocated Memory



- Memory dynamically allocated by a function **not automatically de-allocated (freed)** when the function returns
- Unless explicitly de-allocated, dynamically allocated memory persists until program ends execution
 - Can be very wasteful of memory

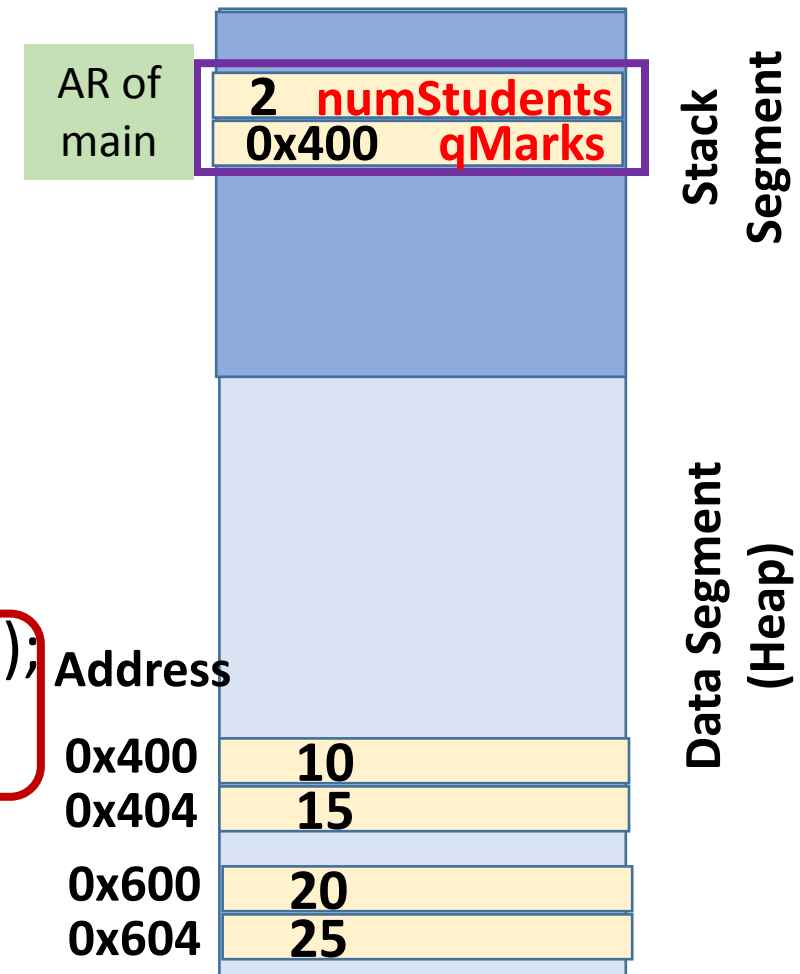
Persistence of Dynamically Allocated Memory

```
int * altReadQuizMarks(int n)
{ int * marks, * temp, i;
  marks = new int[n];
  temp = new int[n];
  if ((marks == NULL) || (temp == NULL))
  { return NULL; }
  for (i = 0; i < n; i++) {
    cin >> marks[i];
    temp[i] = marks[i] + 10;
  }
  // Some computation with temp
  return marks;
}
```



Persistence of Dynamically Allocated Memory

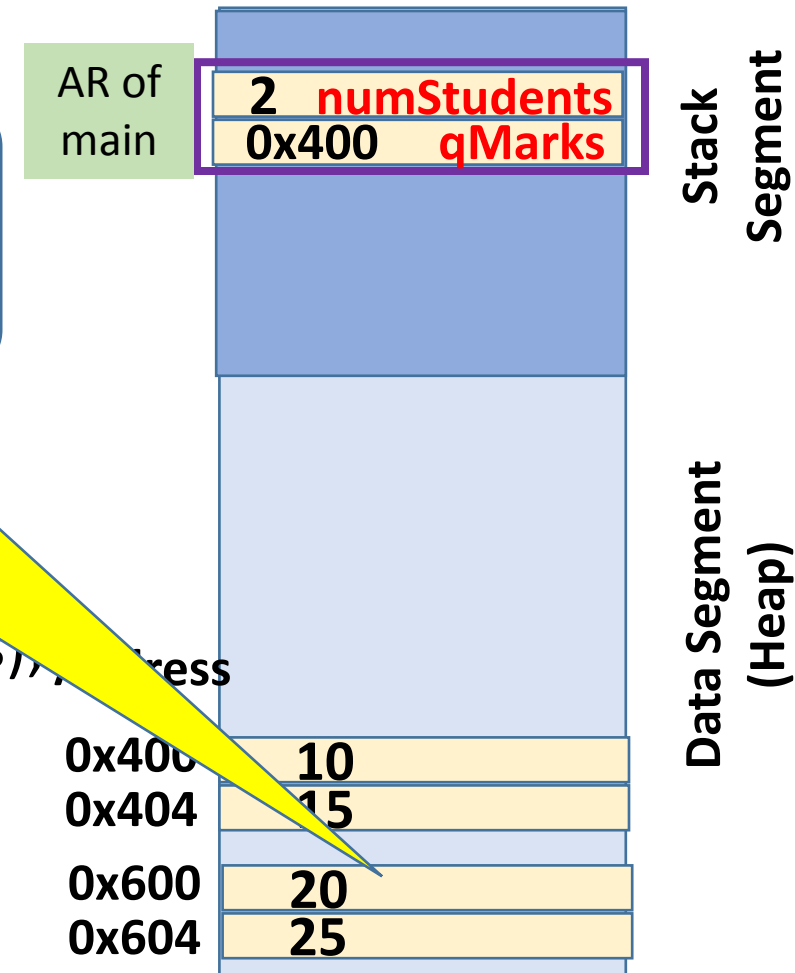
```
int main()
{ int numStudents; int *qMarks;
  cout << "Given student count: " << endl;
  cin >> numStudents;
  cout << "Give marks of students" << endl;
  qMarks = altReadQuizMarks(numStudents);
  // Print max and min marks from qMarks
  return 0;
}
```



Persistence of Dynamically Allocated Memory

```
int main()
{
    int numStudents;
    cout << "Enter number of students: ";
    cin >> numStudents;
    cout << "Give marks of students" << endl;
    qMarks = altReadQuizMarks(numStudents);
    // Print max and min marks from qMarks
    return 0;
}
```

**Array temp of no use, but
persists in heap.
Wasted memory space.**



Explicitly De-allocating Dynamic Memory



- C++ provides a special construct to explicitly de-allocate dynamically allocated memory

Allocation: `T * myVar;`
`myVar = new T;`

De-allocation: `delete myVar;`

Allocation: `T * myArray;`
`myArray = new T[n];`

De-allocation: `delete[] myArray;`

Explicitly De-allocating Dynamic Memory

- C++ programmer has to explicitly de-allocate dynamically allocated memory

Note slightly different usage for dynamically allocated variables and arrays

Allocation:

```
myVar = new T;
```

De-allocation: **delete** myVar;

Allocation: `T * myArray;`

```
myArray = new T[n];
```

De-allocation: **delete[]** myArray;

Good Programming Practice



- Always de-allocate dynamically allocated memory once you have no further use of it
- Check for valid address before de-allocating memory

if (myArray != NULL) { delete[] myArray;}

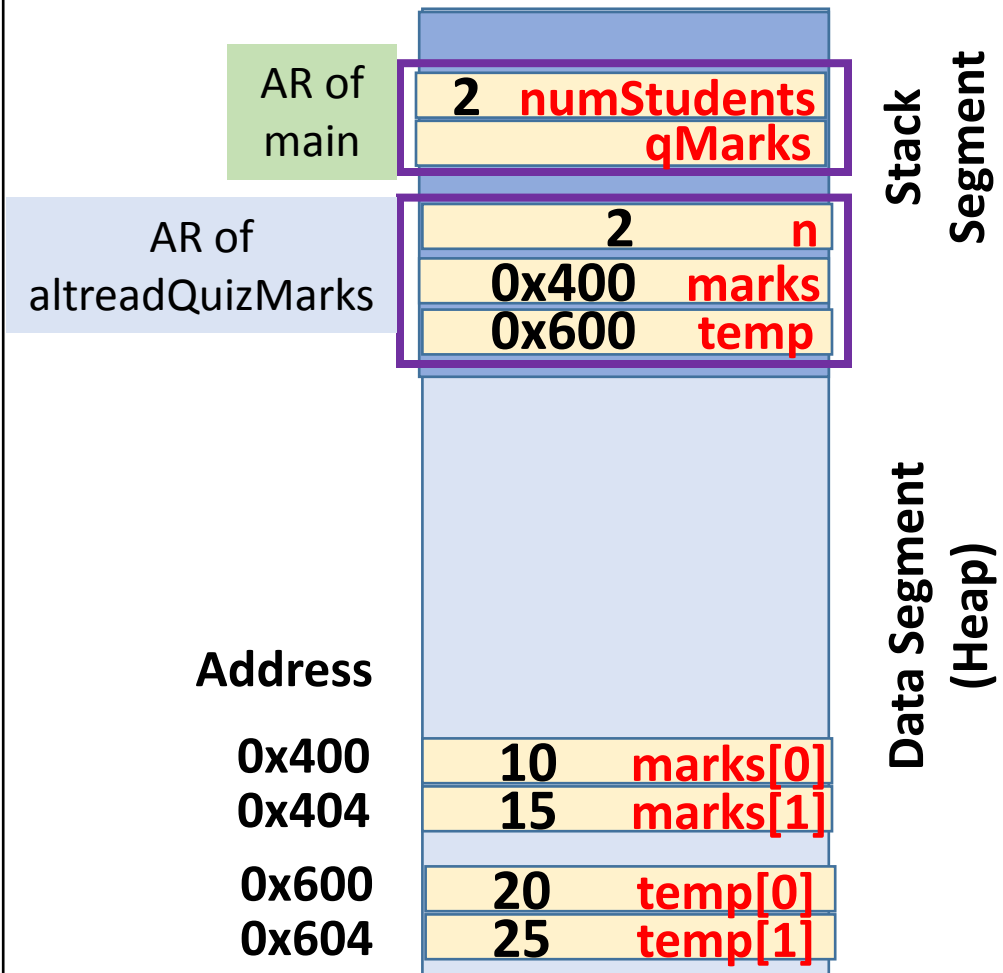
preferred over

delete[] myArray;

- De-allocating memory at address 0x0 (NULL pointer) will cause program to crash

De-allocating Dynamically Allocated Memory

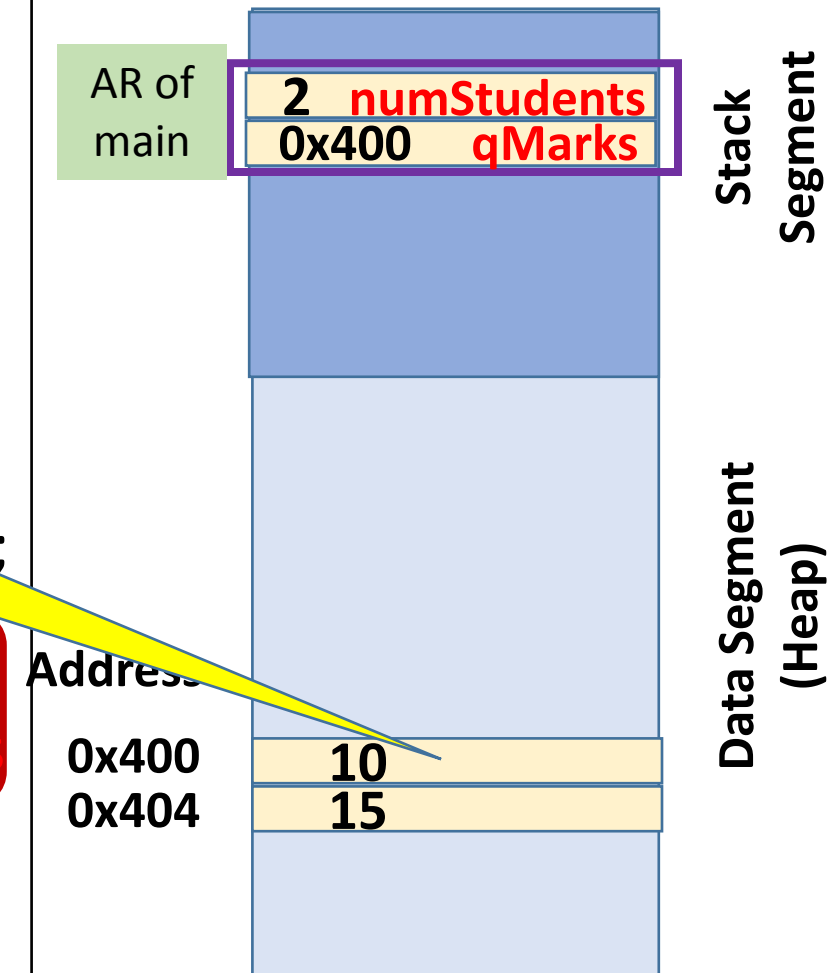
```
int * altReadQuizMarks(int n)
{ int * marks, * temp, i;
  marks = new int[n];
  temp = new int[n];
  if ((marks == NULL) || (temp == NULL)) {
    return NULL; }
  for (i = 0; i < n; i++) {
    cin >> marks[i];
    temp[i] = marks[i] + 10;
  }
  // Some computation with temp
  if (temp != NULL) {delete[] temp; }
  return marks;
}
```



De-allocating Dynamically Allocated Memory

```
int main()
{
    int numStudents;
    cout << "Enter number of students: ";
    cin >> numStudents;
    cout << "Give marks of students: ";
    qMarks = readQuizMarks(numStudents);
    // Print max and min marks from qMarks
    return 0;
}
```

**Only persisting
dynamically allocated
memory is something
that is needed in "main"**



De-allocating Dynamically Allocated Memory

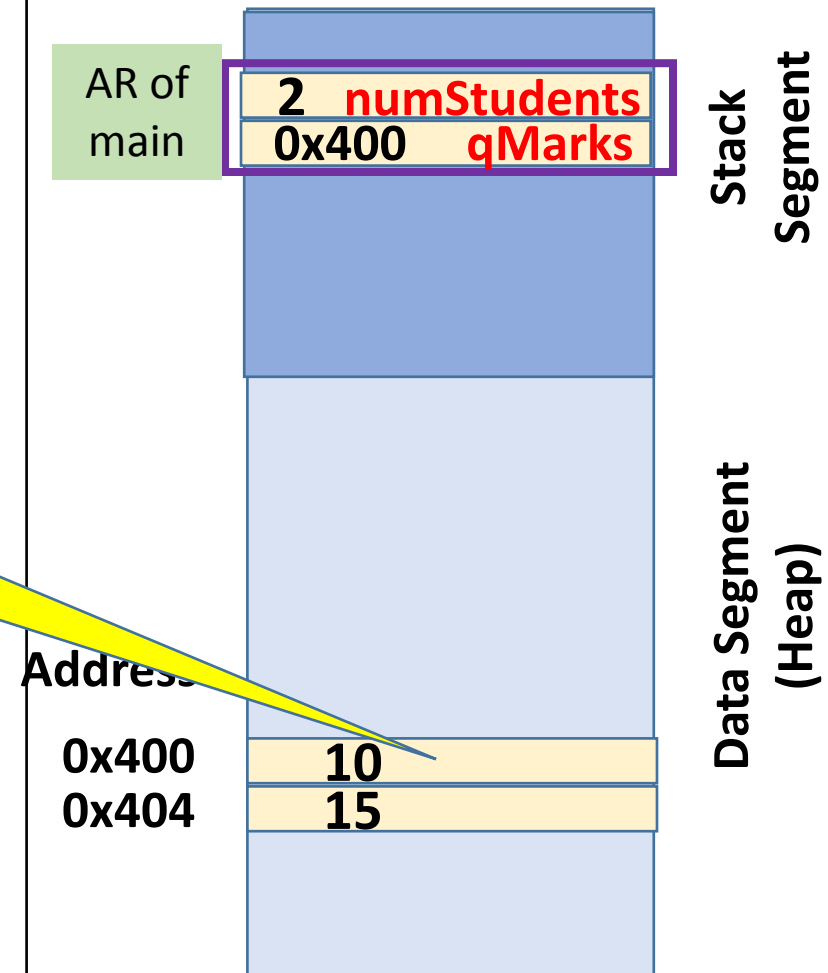
```
int main()
{
    int numStudents;
    cout << "Enter number of students: ";
    cin >> numStudents;

    cout << "Give marks of students: ";
    qMarks = readQuizMarks(numStudents);

    // Print max and min marks from qMarks
    if (qMarks != NULL) { delete[] qMarks; }

    return 0;
}
```

**“main” must now
de-allocate this memory
dynamically allocated by
altReadQuizMarks**



Summary



- Persistence of dynamically allocated memory
- Need for explicit de-allocation of dynamically allocated memory
- “delete” construct in C++
- Good programming practices when de-allocating dynamic memory