**CSE 351**

**Programming Languages**

**Homework Assignment #4**

**Due Date: 22/05/2020 @17:00**

**1. Consider the following program in a C-like language:**

**void Q(m, n) {**

**int k;**

**k=1;**

**n=m+1;**

**m=m+4;**

**print i, j, k, a[1], a[2];**

**}**

**void P(k, w) {**

**int i;**

**i=0;**

**j=k+w;**

**w=j+k;**

**print i, j, k, a[1], a[2];**

**Q(i, a[j]);**

**}**

**void main() {**

**int i, j, k;**

**int a[3];**

**i=1;**

**j=2;**

**k=6;**

**a[1]=5;**

**a[2]=1;**

**P(i, a[j]);**

**}**

**What will be the output of the program if dynamic scoping is used and**

**a) All parameters are passed by reference parameter passing method? Please fill this table according to the steps of execution.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **main** | | | | | | **P** | | | **Q** | | |
|  | **i** | **j** | **k** | **a[0]** | **a[1]** | **a[2]** | **k** | **w** | **i** | **m** | **n** | **k** |
| **main** | 1 | 2 | 6 | 0 | 5 | 1 | Null | Null | Null | Null | Null | Null |
| **P** | 1 | 2 | 6 | 0 | 5 | 1 | 1 | 1 | 0 | Null | Null | Null |
| **j=k+w** | 1 | 2 | 6 | 0 | 5 | 1 | 1 | 1 | 0 |  |  |  |
| **w=j+k** | 1 | 2 | 6 | 0 | 5 | 3 | 1 | 3 | 0 |  |  |  |
| **print in P** | 0, 2, 1, 5 ,3 | | | | | | | | | | | |
| **Q** | 1 | 2 | 6 | 0 | 5 | 3 | 1 | 3 | 0 | \*p.i=0 | \*a[2]=3 | 1 |
| **n=m+1** | 1 | 2 | 6 | 0 | 5 | 1 | 1 | 3 | 0 | 0 | 1 | 1 |
| **m=m+4** | 1 | 2 | 6 | 0 | 5 | 1 | 1 | 3 | 4 | 4 | 1 | 1 |
| **print in Q** | 4, 2,1,5,1 | | | | | | | | | | | |

**b) All parameters are passed by value parameter passing method? Please fill this table according to the steps of execution.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **main** | | | | | | **P** | | | **Q** | | |
|  | **i** | **j** | **k** | **a[0]** | **a[1]** | **a[2]** | **k** | **w** | **i** | **m** | **n** | **k** |
| **main** | 1 | 2 | 6 | 0 | 5 | 1 |  |  |  |  |  |  |
| **P** | 1 | 2 | 6 | 0 | 5 | 1 | 1 | 1 | 0 |  |  |  |
| **j=k+w** | 1 | 2 | 6 | 0 | 5 | 1 | 1 | 1 | 0 |  |  |  |
| **w=j+k** | 1 | 2 | 6 | 0 | 5 | 1 | 1 | 3 | 0 |  |  |  |
| **print in P** | 0,2,6,5,1 | | | | | | | | | | | |
| **Q** | 1 | 2 | 6 | 0 | 5 | 1 | 1 | 3 | 0 | 0 | 1 | 1 |
| **n=m+1** | 1 | 2 | 6 | 0 | 5 | 1 | 1 | 3 | 0 | 0 | 1 | 1 |
| **m=m+4** | 1 | 2 | 6 | 0 | 5 | 1 | 1 | 3 | 0 | 4 | 1 | 1 |
| **print in Q** | 0,2,1,5,1 | | | | | | | | | | | |

**c) All parameters are passed by value-result parameter passing method? Please fill this table according to the steps of execution.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **main** | | | | | | **P** | | | **Q** | | |
|  | **i** | **j** | **k** | **a[0]** | **a[1]** | **a[2]** | **k** | **w** | **i** | **m** | **n** | **k** |
| **main** | 1 | 2 | 6 | 0 | 5 | 1 |  |  |  |  |  |  |
| **P** | 1 | 2 | 6 | 0 | 5 | 1 | 1 | 1 | 0 |  |  |  |
| **j=k+w** | 1 | 2 | 6 | 0 | 5 | 1 | 1 | 1 | 0 |  |  |  |
| **w=j+k** | 1 | 2 | 6 | 0 | 5 | 1 | 1 | 3 | 0 |  |  |  |
| **print in P** | 0,2,6,5,1 | | | | | | | | | | | |
| **Q** | 1 | 2 | 6 | 0 | 5 | 1 | 1 | 3 | 0 | 0 | 1 | 1 |
| **n=m+1** | 1 | 2 | 6 | 0 | 5 | 1 | 1 | 3 | 0 | 0 | 1 | 1 |
| **m=m+4** | 1 | 2 | 6 | 0 | 5 | 1 | 1 | 3 | 0 | 4 | 1 | 1 |
| **print in Q** | 0,2,1,5,1 | | | | | | | | | | | |

**2. What will be the content of the Activation Record stack while running function Q in the previous question? Draw the stack including all the details of the data and control information.**

AR FOR P

forQ

|  |  |
| --- | --- |
| Local | k |
| Parameter | n |
| Parameter | m |
| Dynamic link |  |
| Return | P |
| local | i |
| paramater | w |
| paramater | k |
| Dynamic link | To local |
| return | main |
| local | a[2] |
| local | a[1] |
| local | a[0] |
| local | k |
| local | j |
| Local | i |

MAIN

**3. What will be the content of the 2D stack structure while running function Q, if we apply shallow access method for implementing the dynamic scoping?**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | q |  |  |  |  |  |  |
| p |  | p |  |  |  |  |  |  |
| main | main | main | main | main | main | p | q | q |
| **i** | **j** | **k** | **a[0]** | **a[1]** | **a[2]** | **w** | **m** | **n** |

**4. When you observe several applications using matrices, you found that a matrix with the following configuration is quite commonly used. In such a matrix, all the boundary cells (grey-colored cells) contain data, but the inner-cells are all zeroes.**

**If you store this special type matrix as an ordinary 2D matrix in your programming language, you need to allocate MAX\_ROW\*MAX\_COLUMN\*E bytes (In this example, MAX\_ROW is 8, MAX\_COLUMN is 16 and E is the element size, i.e. if all cells contain bytes E is 1; if all cells contain integers E is 4, etc.) in the memory. You think this is waste of storage space, and want to allocate space for only boundary cells, since the other cells contain all zeroes.**

**Propose an array access function for storing such 2D arrays in a storage-friendly way in memory.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 1 | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 22 |
| 2 | 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 23 |
| 3 | 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 24 |
| 4 | 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 25 |
| 5 | 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 26 |
| 6 | 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 27 |
| 7 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |  |  |  |  |  |  |  | 43 |

Set formula

SET FUNCTION

// ROW = MAX\_ROW

// COLUMN=MAX\_COLUMN

// İ INDICATES FIRST PARAMATER OF ARRAY

// J INDICATES SECOND PARAMATER OF ARRAY

//VAL INDICATES DATA

void set(unsigned i, unsigned j, unsigned val)

if (0 == 0)

{

location(a)+ j\*E = val

}

else if (i == row - 1)

{

s = column + 2 \* (row - 2) + j;

location(a)+s\*E = val;

}

else

{

j = j == 15 ? 1 : 0;

s = column + (i - 1) \* 2 + j;

location(a)+s\*E = val;

}

**ACCESS FUNCTİON**

int get(unsigned i,unsigned j) {

if (!(i == 0 || i == row - 1) && !(j == 0 || j == column - 1))

{

return 0;

}

else {

int E = sizeof(int);

if (i==0)

{

return a[j]; location(a)+j\*E

}

else if(i==row-1)

{

unsigned s = column + 2 \* (row - 2) + j;

return location(a)+s\*E

}

else

{

j = j == 15 ? 1 : 0;

unsigned s = column + (i - 1) \* 2 + j;

return location(a)+s\*E

}

}

}