|  |  |  |  |
| --- | --- | --- | --- |
| **Select a different topic than those already addressed  or create a substantially different question about the same topic.** | | **ANSWER** | **COMMENTS** |
| #. Choose a Lecture Topic Add your name | Add your multiple choice question text, images, and choice(s) here. | Your answer to your question or leave it blank for others to answer. | Provide feedback for a question without any feedback yet. Include your name |
| **0. Linux file copy**  **by Debra Deppeler** | Which command is correct for copying all **.txt** files  from your *handin* directory to your *current working directory*?  Assume that your handin directory is /p/course/cs354/handin/p1/badger/   1. copy /p/course/cs354/handin/p1/badger/\*.txt . 2. cp /p/course/cs354/handin/p1/badger/\*.txt . 3. cp . /p/course/cs354/handin/p1/badger/\*.txt 4. copy . /p/course/cs354/handin/p1/badger/\*.txt 5. mv /p/course/cs354/handin/p1/badger/\*.txt cwd | B |  |
| **1.Java vs C**  **By Kailun Teng** | Please fill in the blanks.  In C, segmentation fault is caused by a \_\_\_\_\_\_\_\_ an uninitialized pointer, or going past the end of an array, etc. | dereferencing | Michael Ortiz:  This WILL be important on the exam. Make sure to know the other types of errors too, I could easily see a question that asks you what type of error is returned. |
| **2. Coding in C remotely** |  |  |  |
| **3.Build an executable**  **Jianping Shen** | Which command will display warning information during the building process?   1. -m32 2. -warn 3. -Window 4. -Wall 5. -std=gnu99 | D | Chengpo Yan:  It would be extremely important to know the function of the parameters in gcc. |
| **4.C Program Structure**  **Yufei guo** | Sequencing, selection and \_\_\_\_\_\_\_ are the three ways of program control? | repetition | I remember in the outline it’s repetition, but in many other places the word iteration is used. Though they have the same meaning here. |
| **5. C Control Flow** |  |  |  |
| **6. C Pointers** | What is the output after the following pseudo-code is compiled and executed?  *int A;*  *int\* B;*  *int\*\* C;*  *A = 2;*  *B = &A;*  *C = &B;*  *\*\*C = 4;*  *printf("%d, %d, %d", A, \*B, \*\*C);*  1. ) 2, 2, 4  2. ) segmentation fault  3.) 4, 4, 4  4.) 2, (mem add of A), (mem add of B) | 3 | Justin Pihowich:  This question is helpful in recognizing the relationship between pointers and addresses. It also helps to understand that the value of an int is stored at its address. |
| **7. 1D Arrays (SAA)**  **By Kevin Xiao** | Given function  void sampleFunction() {  int x[8];  int \*y = x;  What would be the correct way to assign a value of 46 to the element at index 5?   1. x[5] = 46; 2. (y + 5) = 46; 3. \*(y + 5) = 46; 4. \*x[5] = 46; |  |  |
| **8. Arrays and Pointers**  **By Weihao Xu** | Given an array  Void somefunction(){  int p[10];  int \*q = p;  // code  }  Which of the following code is valid to assign value 15 at the index of 1   1. \*p = 15; 2. \*(q+1) = 15; 3. p+1=15; 4. I only 5. II only 6. I and III 7. II and III | B | Israel Kwilinski:  This is a great question that reminds us of the correct syntax to assign values at certain indexes in arrays using address arithmetic. Very likely topic to appear on the exam at least once. |
| **9.Passing Arrays**  **By Yiyang Gu** | void f1(int(\*p)[4]){  return 0;  }  int a[4]={0, 1, 2, 3};  int b[3][4]={{0, 1, 2, 3}, {4, 5, 6, 7}, {8, 9, 10, 11}};  Which of the following is an invalid statement of passing an array to f1?  A. f1(b);  B. f1(&a);  C. f1(b[1]);  D. f1(&b[1]); | C | Runxin Gao  This question reminds me of a question on HW1. Passing arrays, especially 2d arrays, is tricky sometimes. |
| **10. Arrays on Heap**  **By Runxin Gao** | Consider the code below:  int main(){  int \*p=malloc(sizeof(int));  p=malloc(sizeof(int));  free(p);  }  Which of the following statements about the program above are true?  1) There is a memory leak  2) There is a compiler error  3) There will be a segmentation fault during the execution  A: 1) only  B: 1)and 3) only  C: 2) and 3)  D: None are true | A |  |
| **11. C strings**  **by Israel Kwilinski** | Consider the code below:  #include <stdio.h>  int main() {  char \*str = "foo";  char greeting[6] = "hello";  str = "hi";  greeting = "world";  printf("%s\n", str);  return 0;  }  What statement most accurately describes what happens when main() is executed?   1. A buffer overload caused by: char greeting[6] = “hello” 2. An assignment error caused by: str = “hi”; 3. An assignment error caused by: greeting = “world”; 4. No errors. The output of the program is “hi” followed by a newline character. | C | Yiyang Gu:  This question reminds me that in order to copy a string to a char array, we have to use strcpy. |
| **12.string.h functions** | Which is the correct program output?  #include <stdio.h>  #include<string.h>    int main(void)  {  char s1[15] = "Madtown";  char s2[] = "Badgers";  char \* s3 = strcat(s1, s2);  printf("s1 = %s, s3 = %s, compare their address = %d\n", s1, s3, s3 == s1);  return 0;  }  A. s1 = MadtownBadgers, s3 = MadtownBadgers, compare their address = 1  B. s1 = Madtown, s3 = MadtownBadgers, compare their address = 0  C. s1 = MadtownBadgers, s3 = Madtown, compare their address = 0 | A | Jianping Shen:  This question reminds me that strcat() doesn’t make a new address and copy everything in s1, s2. It only copies s2 following s1 if it s1 has enough space and returns the address of s1. |
| **13. 2D Arrays on Heap by Jorge Tenorio** | Which of the following are equivalent to m[i][j]?   1. \*(m[ i ] + j) 2. (\*(m + i))[ j ] 3. \*(\*(m + i) + j) 4. I only 5. II and III 6. III only 7. All are equivalent. | D |  |
| **14. 2D Arrays on Stack**  **By Chengpo Yan** | Suppose we have the following code:  int arr[354][354];  someFunction(arr);  Which of the following function declarations is correct?  I. someFunction(int (\*a)[354])  II. someFunction(int \*a[354])  III. someFunction(int a[3090][355])  IV. someFunction(int a[][354])  V. someFunction(int \*\*a)  A. I & III  B. I & IV  C. V only  D. I & II & III | B | Jinming:  This question remind me about that if the declaration is an array, can’t use pointer to substitute the parameter. |
| **15. Arrays: Stack vs Heap by Yingze Hou** | Suppose there are two 3 by 3 2-d array allocated on Heap and Stack, with same content:  1,2,3  4,5,6  7,8,9  The var name for SAA is m, and the pointer name for Heap allocated is ptr.  Which of the following way is correct for getting 5 if the array is SAA:   1. \*(m[1]+1) 2. \*(m+4) 3. \*(\*m+3\*1+1) 4. \*(\*(m+1)+1)   Which of the following way is correct for getting 5 if the array is allocated in Heap:   1. ptr[4] 2. \*\*(ptr+4) 3. \*(\*ptr+3\*1+1) 4. \*(\*(ptr+1)+1) | C,D | Zhan Yu: This question reminds me that if the array is SAA and the stack frame blocks are continuous for allocating array elements , we can use C which gets the element by “climbing up” the stack frame with 4 bytes each time. |
| **16. Array Caveats**  **Justin Pihowich** | int even[1][3] = {2, 4, 6, 8, 10}; // SAA array  doSomethingWithArray(even, 1, 3);  // size of 2D array as last 2 arguments  Given this function, which of the following are not type compatible and produce a compiler error with the array even?   1. void showArray(int even[1][3], …) 2. void showArray(int even[][3], …) 3. void showArray(int even[1][5], …) 4. void showArray(int even[4][3], …) | C |  |
| **17. CLAs and PAs**  **Haoming Meng** | What is the third CLA of the prompt below?  **Command line:**  $gcc myprog.c -Wall -m32 -std=gnu99   1. gcc 2. -m32 3. c 4. -Wall | D |  |
| **18. Structs**  **By Wenshuo Xie** | Choose wrong statements about C structure.?    int main()  {  struct ship  {    };  return 0;  }  A) It is wrong to define an empty structure.  B) Member variables can be added to a structure even after its first definition.  C) There is no use of defining an empty structure.  D) Structure elements can be initialized at the time of declaration. | A B D |  |
| **19. Passing Structs w/o ptr** |  |  |  |
| **20. Pointers to Structs** |  |  |  |
| **21. Standard I/O** |  |  |  |
| **22. String I/O** |  |  |  |
| **23. File I/O** |  |  |  |
| **24. Copying text files** |  |  |  |
| **25. Virtual Address Space (IA-32/Linux)** |  |  |  |
| **26. Three Faces of Memory**  **by Zhan Yu** | Choose the option that fills the blank correctly.  The amount of Physical Memory Space is \_\_\_ than that of virtual address space. The CPU has the \_\_\_ access and the \_\_ size of storage.  A.bigger, slowest, relatively smaller  B.smaller, fastest, relatively smaller  C.bigger, fastest, relatively bigger  D.smaller, slowest, relatively smaller | B |  |
| **27. Linux Processes and Address Spaces** |  |  |  |
| **28. p1 encode/decode**  **By Jinming Zhang** | Which of the following should be put into the <option>?  Assume we are in the assembling phase:  gcc <option> decode.c -Wall -m32 -std=gnu99   1. -c 2. -E 3. -o 4. -S | A |  |
| **29. p2A n\_in\_a\_row**  **By Hongru Zhou** | What is the most appropriate way to free the memory allocated below?  int \*\* m = malloc(sizeof(int \*) \* n)  for (int i = 0; i < n; i++) {  \*(m + i) = malloc(sizeof(int)) \* n)  }   1. free(m); 2. for (int i = 0; i <n; i++)  free(\*(m + i));   free (m);   1. free(m);   for (int i = 0; i <n; i++)  free(\*(m + i));    D) m = NULL;  E) for (int i = 0; i <n; i++) {  free(\*(m + i));  \*(m + i) = NULL;  }  free(m);  m = NULL; | E | Hongru Zhou:  We need to free the memory in the reverse order of how it is allocated. |
| **30. p2B magic square** |  |  |  |
| **31.** |  |  |  |
| **32.** |  |  |  |
| **33.** |  |  |  |
| **34.** |  |  |  |
| **35.** |  |  |  |