| Question No. 01 |
| --- |
| Explain the pass by value and pass by reference mechanisms. Give examples that show their difference. |
| **When we call a function, we can send arguments using two ways. We can pass the value of a variable (by writing the name of the variable) and we can pass the address of a variable (by writing & with the name of the variable) which is called “pass by reference”.**  **When we use pass by value: the original variables do not get changed. We merely create a copy of that variable in our user defined function and use it.**  **When we use pass by reference: the address of the variable is passed and is saved in a pointer. So whatever changes we make, it happens to the original variable.**  **Example:**  **#include <stdio.h>**  **void swap\_by\_reference(int \*x, int \*y){**  **int temp = \*x;**  **\*x = \*y;**  **\*y = temp;**  **}**  **void swap\_by\_value(int x, int y){**  **int temp = x;**  **x = y;**  **y = temp;**  **}**  **int main(){**  **int a = 10, b = 20;**  **printf("Before: %d %d\n", a, b);**  **swap\_by\_value(a, b);**  **printf("After: %d %d\n", a, b);**  **a = 5, b = 7;**  **printf("Before: %d %d\n", a, b);**  **swap\_by\_reference(&a, &b);**  **printf("After: %d %d\n", a, b);**  **return 0;**  **}** |

| Question No. 02 |
| --- |
| Consider the function -  int f(int n, int a[]) {  Int cnt = 0;  for (int i=0; i<n; i++) {  if (a[i] == a[0]) cnt++;  }  return cnt;  }  Explain what it does in one sentence. What is the return value when n = 5 and a = {1, 2, 1, 2, 1}? |
| **It counts the number of array elements that are equal to the first element of the array. The return value will be 3.** |

| Question No. 03 |
| --- |
| Implement the makeStrCopy function. Remember that, It takes a string in copies to an output string out. The signature should be void makeStrCopy(char in[], char out[]). For example - if in = “hello”, after calling makeStrCopy, out should also be “hello” |
| **void makeStrCopy(char in[], char out[])**  **{**  **while (\*in != '\0')**  **{**  **\*out = \*in;**  **out++;**  **in++;**  **}**  **\*out = '\0';**  **}** |

| Question No. 04 |
| --- |
| Dynamically allocate an array of floats with 100 elements. How much memory does it take? |
| **float \*ptr = (float \*)malloc(100 \* sizeof(float));**  **It takes 400 bytes of memory.** |

| Question No. 05 |
| --- |
| Suppose int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9}. Suppose the address of a[0] is at 6000. Find the value of the following -  a. a[8]  b. &a[5]  c. a  d. a+4  e. \*(a+2)  f. &\*(a+4) |
| 1. a[8] = 9 2. &a[5] = 6020 3. a = 6000 4. a+4 = 6016 5. \*(a+2) = 3 6. &\*(a+4) = 6016 |

| Question No. 06 |
| --- |
| Ash tries to implement bubble sort the following way. In particular, notice that the loop iterates on the array in reverse. Fill in the box to implement the function.  void sort(int n, int a[]) {  for (int steps=0; steps<n; steps++) {  for (int i=n-1; i>0; i--) {  ///Write code here  }  }  } |
| void sort(int n, int a[]) {  for (int steps=0; steps<n; steps++) {  for (int i=n-1; i>0; i--) {    if (array[i] < array[i - 1])  {  int temp = array[i];  array[i] = array[i - 1];  array[i - 1] = temp;  }  }  }  } |

| Question No. 07 |
| --- |
| implement the is\_reverese\_sorted() function to check if an array is reverse sorted. For example if a = {6, 4, 3, 1}. Then is\_reverse\_sorted should return True |
| **bool \_reverse\_sorted(int arr[], int n)**  **{**  **if (n == 0 || n == 1)**  **return true;**  **for (int i = 1; i < n; i++)**  **if (arr[i - 1] < arr[i])**  **return false;**  **return true;**  **}** |

| Question No. 08 |
| --- |
| Modify the Selection sort function so that it sorts the array in reverse sorted order, ie. from the largest to smallest. For example reverse sorting a = {3, 4, 2, 5, 1} should result in {5, 4, 3, 2, 1}. Use the is\_reverse\_sorted() function to break early from the function if the array is already sorted |
| **#include <stdio.h>**  **#include <stdbool.h>**  **void swap(int \*x, int \*y){**  **int temp = \*x;**  **\*x = \*y;**  **\*y = temp;**  **}**  **bool \_reverse\_sorted(int arr[], int n){**  **if (n == 0 || n == 1)**  **return true;**  **for (int i = 1; i < n; i++)**  **if (arr[i - 1] < arr[i])**  **return false;**  **return true;**  **}**  **void selectionSort(int array[], int size){**  **for (int step = 0; step < size - 1; step++){**  **int min\_idx = step;**  **for (int i = step + 1; i < size; i++)**  **if (array[i] > array[min\_idx])**  **min\_idx = i;**  **swap(&array[min\_idx], &array[step]);**  **if (\_reverse\_sorted(array, size))**  **break;**  **}**  **}**  **void printArray(int array[], int size){**  **for (int i = 0; i < size; ++i)**  **printf("%d ", array[i]);**  **printf("\n");**  **}**  **int main(){**  **int data[] = {3, 4, 2, 5, 1};**  **int size = sizeof(data) / sizeof(data[0]);**  **selectionSort(data, size);**  **printf("Sorted Array in Descending Order: ");**  **printArray(data, size);**  **}** |

| Question No. 09 |
| --- |
| We wrote a program to find all positions of a character in a string with the strchr  function. Now do the same without using strchr |
| **#include <stdio.h>**  **#include <string.h>**  **char \*ownstrchr(char \*s, char c)**  **{**  **while (\*s != c && \*s != '\0')**  **s++;**  **if (\*s == c)**  **return s;**  **else**  **return NULL;**  **}**  **int main()**  **{**  **char text[1001];**  **gets(text);**  **char pattern;**  **scanf(" %c", &pattern);**  **char \*cur = text;**  **while (cur){**  **char \*pos = ownstrchr(cur, pattern);**  **if (pos == NULL)**  **break;**  **printf("Found at %d\n", pos - text);**  **cur = pos + 1;**  **}**  **return 0;**  **}** |

| Question No. 10 |
| --- |
| Is there any difference in output if you call strstr(text, “a”) and strchr(text,  ‘a’)? Explain with examples. |
| **strstr searches for a specific substring and strchr searches for a specific character. For the given scenario, the output will be the same as the substring contains a single character.**  **#include <stdio.h>**  **#include <string.h>**  **int main(){**  **char text[1001];**  **gets(text);**  **char \*pos = strstr(text, "a");**  **if (!pos) printf("NOT FOUND\n");**  **else printf("Found at %d\n", pos - text);**  **return 0;**  **}**  **#include <stdio.h>**  **#include <string.h>**  **int main(){**  **char text[1001];**  **gets(text);**  **char \*pos = strchr(text, 'a');**  **if (!pos) printf("NOT FOUND\n");**  **else printf("Found at %d\n", pos - text);**  **return 0;**  **}** |

**Question Paper:** [Exam - 05](https://docs.google.com/document/d/1H1eMiaysK0JbD84C89usJKR8NwB3KxfZOtutWtB8m9M/edit)