**Answer Script of Exam - 05**

|  |
| --- |
| Question No. 01 |
| Explain the pass by value and pass by reference mechanisms. Give examples that show their difference. |
| In pass by value, the value of a function parameter is copied to another location of the memory. When accessing or modifying the variable within the function, it accesses only the copy. Thus, there is no effect on the original value.  #include<stdio.h>  int findNewValue(int val);  int main()  {  int value= 5;  int newValue= findNewValue(value);  printf("New Value: %d\n", newValue);  return 0;  }  int findNewValue(int val)  {  val+5;  }  In the program illustrated above, the variable value stores integer 5. The findNewValue is a function. The value is passed to that function. In the function, the value is then copied to a new memory location called newValue. The function then returns an integer. And, this integer is stored in the newValue variable of the main function. Finally, the newValue is printed on the console.  In the function, the value copied to a new memory location is called newValue. The changes are made to that newValue, not to the original value. This method is called Pass by Value.  In pass by reference, the memory address is passed to that function. In other words, the function gets access to the actual variable. An example is as follows.  The variable value stores integer 5. The findNewValue is a function. The address of the memory location ‘value’ is passed to that function. Thus, the function gets this value. The newValue is a pointer. It points to the original memory location called value. The function adds 5 to the original value pointed by newValue. Then, the calculated value is returned and stored into the newValue variable. Finally, the newValue is printed on the console.  In this method, the memory location passes to the function. Therefore, the changes are made to the original value. Hence, this method is called Pass by Reference. |

|  |
| --- |
| 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}? |
| **Ans:** The function returns the number of values same as the first value in the array.  If we pass the array size n=5 and the values of the array a[]={1, 2, 1, 2, 1} to the function, it will return 3 as the first value (the value at index 0 or a[0] = 1) repeats two more times at index 2 or a[2] and index 4 or a[4]. |

|  |
| --- |
| 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” |
| **Ans:**  #include<stdio.h>  #include<string.h>  void makeStrCopy (char in[], char out[]);  int main()  {  char in[100], out[100];  gets(in);  makeStrCopy(in, out);  printf("%s\n", out);  return 0;  }  void makeStrCopy (char in[], char out[])  {  int len= strlen(in);  for(int i=0; i<len; i++)  {  out[i]= in[i];  }  } |

|  |
| --- |
| Question No. 04 |
| Dynamically allocate an array of floats with 100 elements. How much memory does it take? |
| **Ans:**  float \*array= (float \*) malloc(100\*sizeof(float));  As the size of float is 4 bytes, the array will allocate 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) |
| **Ans:**  If the array is supposed as int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};  And if the address of a[0] or &a[0] is supposed to be 6000,   1. The value of a[8] will be 9. 2. The value of &a[5] will be 6020. (&a[5]= address of index a[5]= address of a[0] + 5\*sizeof(int)) 3. The value of a will be 6000. (array name is a pointer type variable that contains the address of the array which is same as the address of the first element of that array) 4. The value of a+4 will be 6016. (a+4= address a + 4\*sizeof(int)) 5. The value of \*(a+2) will be 3. (\*(a+2)= content of address a+2 = a[2]) 6. The value of &\*(a+4) will be 6016. (&\*(a+4)= address of content of address a+4 = address a+4) |

|  |
| --- |
| 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         }     }  } |
| **Ans:**  if (a[i-1] > a[i])  {  int temp = a[i-1];  a[i-1] = a[i];  a[i] = temp;  } |

|  |
| --- |
| Question No. 07 |
| implement the is\_reverese\_sorted() function to check if an array reverse sorted. For example if a = {6, 4, 3, 1}. Then is\_reverse\_sorted should return True |
| **Ans:**  #include<stdio.h>  #include<stdbool.h>  bool is\_reverese\_sorted (int n, int a[]);  int main()  {  int n;  scanf("%d", &n);  int a[n];  for(int i=0; i<n; i++)  {  scanf("%d", &a[i]);  }  printf("%s", is\_reverese\_sorted (n, a)? "True":"False");  return 0;  }  bool is\_reverese\_sorted (int n, int a[])  {  for (int i=0; i<n-1; i++)  {  if (a[i] < a[i+1])  {  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 |
| **Ans:**  #include<stdio.h>  #include<stdbool.h>  bool is\_reverese\_sorted(int n, int a[]);  void sort(int n, int a[]);  int main()  {  int n;  scanf("%d", &n);  int a[n];  for(int i=0; i<n; i++)  {  scanf("%d", &a[i]);  }  sort(n, a);  for(int i=0; i<n; i++)  {  printf("%d", a[i]);  if(i != n-1)  {  printf(" ");  }  }  return 0;  }  void sort(int n, int a[])  {  if(is\_reverese\_sorted(n, a))  {  return;  }  else  {  int temp;  for(int j=0; j<n; j++)  {  for(int k=0; k<n-1; k++)  {  if(a[k]<a[k+1])  {  temp=a[k];  a[k]=a[k+1];  a[k+1]=temp;  }  }  }  }  }  bool is\_reverese\_sorted(int n, int a[])  {  for(int i=0; i<n-1; i++)  {  if(a[i]<a[i+1])  {  return false;  }  }  return true;  } |

|  |
| --- |
| 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 |
| **Ans:**  #include<stdio.h>  #include<string.h>  int main()  {  char c, str[100];  gets(str);  scanf("%c", &c);  int flag=0;  int len= strlen(str);  printf("\nPositions of the character in the given string are: ");  for(int i=0; i<len; i++)  {  if(str[i] == c)  {  flag= 1;  printf("%d ", i+1);  }  }  if (flag==0)  {  printf("None!\nThe character is not present in the given string.");  }  return 0;  } |

|  |
| --- |
| Question No. 10 |
| Is there any difference in output if you call strstr(text, “a”) and strchr(text,  ‘a’)? Explain with examples. |
| **Ans:**  No, there will be no differencein output if we call strstr(text, “a”) and strchr(text, ‘a’).  We know that in strstr(text, “abc”); the function strstr searches the string named *text* to see whether the sub-string *abc* is contained in the string *text*. If yes, the function returns the position of the first occurrence of the sub-string. Otherwise, it returns NULL.  Whereas, in strchr(text, ‘a’); the function strchr searches the string named *text* to see whether the character *a* is contained in the string *text*. If yes, the function returns the position of the first occurrence of the character. Otherwise, it returns NULL.  That means, in strstr(text, “a”); the function strstr searches the value *a* as a sub-string and in strchr(text, ‘a’); the function strchr searches the value *a* as a character to see whether the sub-string *a* or the character *a* is contained in the string *text.* As the value of substring “a” and character ‘a’ is same; therefore, there will be no difference in output of the two function statements.  For example:  #include<stdio.h>  #include<string.h>  int main()  {  char text[100];  gets(text);  if(strstr(text, "a")== NULL)  printf("The substring \"a\" is not found in the text.\n");  else  printf("The substring \"a\" is found in the text.\n");  if(strchr(text, 'a')== NULL)  printf("The character 'a' is not found in the text.\n");  else  printf("The character 'a' is found in the text.\n");  return 0;  }  In the above program, if the input is given as “Cumilla”, the output will be  The substring "a" is found in the text.  The character 'a' is found in the text.  And, if the input is given as “Feni”, the output will be  The substring "a" is not found in the text.  The character 'a' is not found in the text. |

—--------------------------------- **END** -----------------------------------------