**Sri Sri University, Cuttack, Odisha.**

**Faculty of Science, Department of Computer Science.**

|  |  |
| --- | --- |
| **Program: B.Sc. – Computer Science, Data Science, & Environmental Science**  **(2020-23 Batch)**  **Subject Code/Subject Name: Data Structure Laboratory**  **Assignment –II** | |
| **Full Name of the Student:** | VINAYAK SANJAY CHAVAN(CS) |
| **Full Roll Number:** | BCS-011 |
| **Program:** | B.Sc. (Computer Sc.) / B.Sc. (Data Sc.) / B.Sc. (Env. Sc.) |
| **Date:** | 27th February, 2021 (10.00 AM – 12.00 Noon) |
| **Signature** |  |

|  |  |
| --- | --- |
| **All Questions are compulsory** | **Total Marks: 60** |

|  |  |
| --- | --- |
| **Question (s)** | **Maximum Marks** |
| 1. Write a program to test dangling pointers by returning the address of a local variable from a function.   **Your code:**  #include <iostream>  int \*dangle(int);  using namespace std;  /\* Write a program to test dangling pointers by  returning the address of a local variable from a function. \*/  int main(void){  int n=100;  int \*ptr;  ptr=dangle(n);    return 0;  }  int \*dangle(int num){  int temp;  temp=num\*2;    cout<<"value of temp is: "<<temp;    return &temp;  }  **Screenshot of output:** | 10 |
| 1. Write a program to access a normal variable “var” with \*\*\*p , \*\*q and \*r.   **Your code:** #include <iostream>  #include <cmath>  using namespace std;  int main(){  // program to access a normal variable "var" with \*\*\*p , \*\*q and \*r.  int var,\*\*\*p,\*\*q,\*r;    cout<<"enter the value of the var: ";  cin>>var;  r = &var;  q = &r;  p = &q;  cout<<"address of var: "<<&var<<endl;  cout<<"var using pointer: "<<\*r<<endl;  cout<<"address of r: "<<&r<<endl;  cout<<"ptr to ptr: "<<\*\*q<<endl;  cout<<"address of q: "<<&q<<endl;  cout<<"ptr to ptr to ptr: "<<\*\*\*p<<endl;  cout<<"address of p: "<<&p<<endl;    cout<<"value of r: "<<\*r<<endl;  cout<<"value of q: "<<\*q<<endl;  cout<<"value of p: "<<\*p<<endl;    return 0;  }  **Screenshot of output:** | 10 |
| 1. Write a function that returns a pointer to the maximum value of an array of floating point data:   ***double\* maxvalue(double\* a, int size)***  If the input size is 0, return NULL.  **Your code:** **#include <iostream>**  **using namespace std;**  **/\* Write a function that returns a pointer to the**  **maximum value of an array of floating point data:**  **=>double\* maxvalue(double\* a, int size) \*/**  **double\* maxvalue(double\* arr,int size){**  **double max=arr[0];**  **int result=0;**  **for(int i=0;i<size;i++){**  **if(arr[i]>max){**  **max=arr[i];**  **result=i;**  **}**  **}**  **return arr+result;**  **}**  **int main(){**  **int length;**  **cout<<"Enter the length of the array: ";**  **cin>>length;**  **double myarray[length];**  **double \*res;**  **cout<<"Enter the elements of the array: ";**  **for(int i=0;i<length;i++){**  **cin>>myarray[i];**  **}**  **res=maxvalue(myarray,length);**  **cout<<"The maximum element in the array is: "<<\*res;**  **return 0;**  **}**  **Screenshot of output:** | 10 |
| 1. Dynamically create 3 square matrices of same sizes (a user input), scan elements of the first two matrices, subtract them and store the result in the third matrix. Use only malloc function to allocate memory dynamically.   **Your code:** **/\* Dynamically create 3 square matrices of same sizes (a user input),**  **scan elements of the first two matrices, subtract them and store the**  **result in the third matrix. Use only malloc function**  **to allocate memory dynamically. \*/**    **#include<iostream>**  **#include <cstdlib>**  **using namespace std;**  **int main(){**  **int num\_row, \*matrix\_1, \*matrix\_2, \*matrix\_3;**  **cout<<"Enter number of rows in matrix";**  **cin >> num\_row;**  **matrix\_1 = (int\*) malloc(sizeof(int) \* num\_row\*num\_row);**  **matrix\_2 = (int\*) malloc(sizeof(int) \* num\_row\*num\_row);**  **matrix\_3 = (int\*) malloc(sizeof(int) \* num\_row\*num\_row);**    **cout << "Input the 1st matrix: "; //enter 1st matrix**  **for(int i=0; i < num\_row \* num\_row; i++){**  **cin >> \*(matrix\_1+i);**  **}**  **cout << "Input the 2nd matrix: "; //enter 2nd matrix**  **for(int i=0; i< num\_row \* num\_row; i++){**  **cin >> \*(matrix\_2+i);**  **}**    **for(int i=0; i<num\_row \* num\_row; i++){ //sub of matrix 1 and matrix 2**  **\*(matrix\_3+i) = \*(matrix\_1+i) - \*(matrix\_2+i);**  **}**  **cout << "subtraction of matrix is: " <<endl;**  **for(int i=0; i< num\_row \*num\_row; i++){**  **cout << \*(matrix\_3+i) << " ";**  **if ((i+1)% num\_row==0){**  **cout <<endl;**  **}**  **}**  **}**  **Screenshot of output:** | 10 |
| 1. Explain the following functions with their prototype. 2. **malloc():** void\* malloc(size\_t size)   =>allocate memory and returns void\*   1. **calloc():** void\* calloc(size\_t count, size\_t size)   =>allocate memory and returns void\*, with initialization to each space by zero.    c) **free():** void free(void\* block)  =>free function releases all your memory created by malloc/calloc.   1. **realloc():** void realloc(void \*\_memory, size\_t \_NewSize)   =>reallocates a block of memory that was previously allocated but not yet freed. | 10 |
| 1. Write a global function called **splice()** that “splices” two int arrays together by first allocating memory for a dynamic array with enough room for both int arrays, and then copying the elements from both arrays to the new array, as follows:  * first, the elements of the first array are inserted up to a given position (input) * then the second array is inserted, * then the remainder of the first array is appended.   *Arguments: The two int arrays, their length, and the position at which they are to be spliced.*  *Return value: A pointer to the new array*  **Your code: #include<iostream>**  **#include<cstdlib>**  **using namespace std;**  **void\* splice(int\*,int\*,int,int,int);**  **int main(){**  **int length1,length2,position,\*arr1,\*arr2,\*res;**  **cout<<"Enter the length of the first array: ";**  **cin>>length1;**  **cout<<"Enter the length of the second array: ";**  **cin>>length2;**  **cout<<"Enter the position: ";**  **cin>>position;**  **arr1=(int\*)malloc(sizeof(int)\*length1);**  **arr2=(int\*)malloc(sizeof(int)\*length2);**  **cout<<"The first array is: ";**  **for(int i=0;i<length1;i++){**  **cin>>\*(arr1+i);**  **}**  **cout<<"The second array is: ";**  **for(int i=0;i<length2;i++){**  **cin>>\*(arr2+i);**  **}**  **splice(arr1,arr2,length1,length2,position);**  **return 0;**  **}**  **void\* splice(int \*arr1,int \*arr2,int length1,int length2,int position){**  **int \*res;**  **int len3=length1+length2;**  **res=(int\*)malloc(sizeof(int)\*len3);**  **for(int i=0;i<len3;i++){**  **for(int i=0;i<=position;i++){**  **\*(res+i)=\*(arr1+i);**  **}**  **for(int i=0;i<length2;i++){**  **\*(res+i+position)=\*(arr2+i);**  **}**  **for(int i=position;i<length1;i++){**  **\*(res+i+length2)=\*(arr1+i);**  **}**  **}**  **cout<<"The new reslutant array is: ";**  **for(int i=0;i<len3;i++){**  **cout<<\*(res+i)<<endl;**  **}**  **}**  **Screenshot of output:** | 10 |