USN | | | | | | | | | | |

Internal Assessment Test 1–Nov 2022

Sub:	Data Structures and Applications					SubCode:	21CS32	Branch:	ISE	
Date:	1.12.2022	Duration:	90min's	MaxMarks:	50	Sem/Sec:	III A, B &C		OBE	
Answer any FIVE FULL Questions								MARKS	CO	RBT
1a)	Consider 2 polynomials, $A(x) = 4x^{15}+3x^4+5$ & $B(x) = x^4+10x^2+1$. Represent polynomials using array of structures (show diagrammatically show these 2 polynomials can be stored in 1D array) and also give its C representation.							6	CO1	L2
1b)	Define Data Structures. Explain the various types of data structures with examples.							4	CO1	L1
2	Write a C program to demonstrate inserting, deleting and searching operations on arrays. Support your program with functions.							10	CO1	L3
3a)	Convert the following infix expressions to postfix expressions : 1. (a+(b-c)*(d-e)%f) 2. (a+b) * d – e / (f +a * d) +c 3. ((a / (b – c + d)) * (e – f) * g)							6	CO2	L2
3b)	Describe how to represent 1D and 2D arrays in memory with diagram.							4	CO1	L2
4a)	Discuss the representation of multidimensional arrays with examples.							6	CO1	L2

P.T.O.

USN | | | | | | | | | | |

Internal Assessment Test 1–Nov 2022

Sub:	Data Structures and Applications					SubCode:	21CS32	Branch:	ISE	
Date:	1.12.2022	Duration:	90min's	MaxMarks:	50	Sem/Sec:	III A, B &C		OBE	
Answer any FIVE FULL Questions								MARKS	CO	RBT
1a)	Consider 2 polynomials, $A(x) = 4x^{15}+3x^4+5$ & $B(x) = x^4+10x^2+1$. Represent polynomials using array of structures (show diagrammatically show these 2 polynomials can be stored in 1D array) and also give its C representation.							6	CO1	L2
1b)	Define Data Structures. Explain the various types of data structures with examples.							4	CO1	L1
2	Write a C program to demonstrate inserting, deleting, and searching operations on arrays.							10	CO1	L3
3a)	Convert the following infix expressions to postfix expressions : 1. (a+(b-c)*(d-e)%f) 2. (a+b) * d – e / (f +a * d) +c 3. ((a / (b – c + d)) * (e – f) * g)							6	CO2	L2
3b)	Describe how to represent 1D and 2D arrays in memory with diagram.							4	CO1	L2
4a)	Discuss the representation of multidimensional arrays with examples.							6	CO1	L2

P.T.O.

4b)	Describe self referential structures with an appropriate example.	4	CO1	L2
5	Discuss the concept of dynamic 2D arrays with example. Write a C program to create and display dynamic 2D array.	10	CO1	L3
6a)	What is realloc()? Explain with proper syntax and example program.	6	CO1	L2
6b)	List and explain any 3 applications of stack data structure.	4	CO2	L1

Faculty Signature CCI Signature HOD Signature

4b)	Describe self referential structures with an appropriate example.	4	CO1	L2
5	Discuss the concept of dynamic 2D arrays with example. Write a C program to create and display dynamic 2D array.	10	CO1	L3
6a)	What is realloc()? Explain with proper syntax and example.	6	CO1	L2
6b)	List and explain any 3 applications of stack data structure.	4	CO2	L1

Faculty Signature CCI Signature HOD Signature

Scheme of Evaluation
Internal Assessment Test 1 –DEC 2022

Sub:	Data Structures and Applications							Code:	21CS32
Date:	1.12.2022	Duration:	90mins	Max Marks:	50	Sem:	III	Branch:	ISE

Note: **Answer Any five full questions.**

Question #		Description	Marks Distribution		Max Marks
1	a)	Consider 2 polynomials, $A(x) = 4x^{15}+3x^4+5$ & $B(x) = x^4+10x^2+1$. Represent polynomials using array of structures (show diagrammatically show these 2 polynomials can be stored in 1D array) and also give its C representation. **Diagrammatic representation** **Explanation + C representation**	3M+3M	6M	10M
	b)	Define Data Structures. Explain the various types of data structures with examples **Definition** **Explaining types with examples**	1M+3M	4M	
2		Write a C program to demonstrate inserting, deleting and searching operations on arrays. Support your program with functions. **Program write up with function for each operation**	1M+3X3M	10M	10M
3	a)	Convert the following infix expressions to postfix expressions : 1. (a+(b-c)*(d-e)%f) 2. (a+b) * d – e / (f +a * d) +c 3. ((a / (b – c + d)) * (e – f) * g) **Conversion steps + Obtaining answer**	3X2M	6M	10M

3	b)	Describe how to represent 1D and 2D arrays in memory with diagram. **Representation and Explaining 1D and 2 D arrays**	2M+2M	4M	
4	a)	Discuss the representation of multidimensional arrays with examples. **Representation + Example**	4M 2M	6M	10M
	b)	Describe self referential structures with an appropriate example. **Explanation + Example**	2M+2M	4M	
5		Discuss the concept of dynamic 2D arrays with example. Write a C program to create and display dynamic 2D array. **Explanation + Example Program**	3M+7M	10M	10M

6	a)	What is realloc()? Explain with proper syntax and example program. **Definition + Syntax with explanation +Example program**	1M+2M+ 3M	6M	
6	b)	List and explain any 3 applications of stack data structure. **List + Explanation**	1M+3M	4M	10M

Scheme of EvaluationInternal Assessment Test 1 –DEC 2022

Sub:	Data Structures and Applications							Code:	21CS32
Date:	1.12.2022	Duration:	90mins	Max Marks:	50	Sem:	III	Branch:	ISE

Note: **Answer Any five full questions.**

1a) Consider 2 polynomials, $A(x) = 4x^{15}+3x^4+5$ & $B(x) = x^4+10x^2+1$. Represent polynomials using array of structures (show diagrammatically show these 2 polynomials can be stored in 1D array) and also give its C representation.

Solution:

Consider the two polynomials

$$\mathbf{A(x) = 4x^{15}+3x^4+5}$$
$$\mathbf{B(x) = x^4+10x^2+1}$$

Figure below shows how these polynomials are stored in the array terms. The index of the first term of A and B is given by startA and startB, respectively, finishA and finishB give the index of the last term of A and B respectively. The index of the next free location in the array is given by avail.

For our example, startA = 0, finishA = 1, startB = 2, finishB = 5, and avail =6.

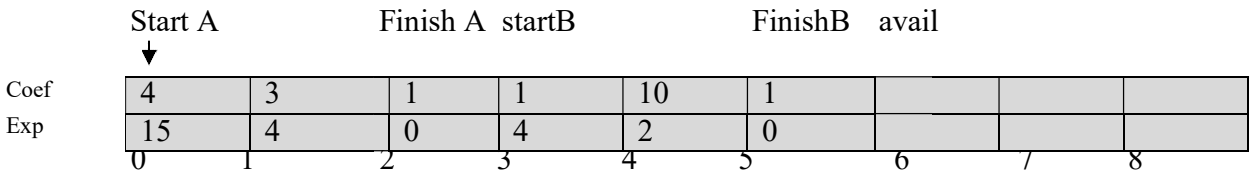


Figure 1.3 Representation of polynomial in array

- There is no limit on the number of polynomials that we can place in terms.
- The total number of nonzero terms must not be greater than MAX_TERMS.

Polynomial Representation in C:

- A polynomial can be represented as an array of structures as follows.
- Only one global array, *terms,* is used to store all the polynomials.
- The C declarations needed are:

```
# define MAX_TERMS 100 /*size of terms array*/
typedef struct {
float coef;
int exp;
} polynomial;
```

```
polynomial terms[MAX_TERMS];
int avail = 0;
```

1b) Define Data Structures. Explain the various types of data structures with examples.
Solution:
Data Structures: A data structure is a particular method of storing and organizing data in a computer so that it can be used efficiently.

The data Structure is classified into
a. **Primitive** data structure: These can be manipulated directly by the machine instructions. Example integer, character, float etc

b. **Non primitive** data structures: They cannot be manipulated directly by the machine instructions. The non primitive data structures are further classified into linear and non linear data structures.
 Linear data structures: show the relationship of adjacency between the elements of the data structures. Example are arrays, stacks, queues , list etc.
 Non linear data structure: They do not show the relationship of adjacency between the elements. Example are Trees and graphs

2. Write a C program to demonstrate inserting, deleting, and searching operations on arrays.
Solution:

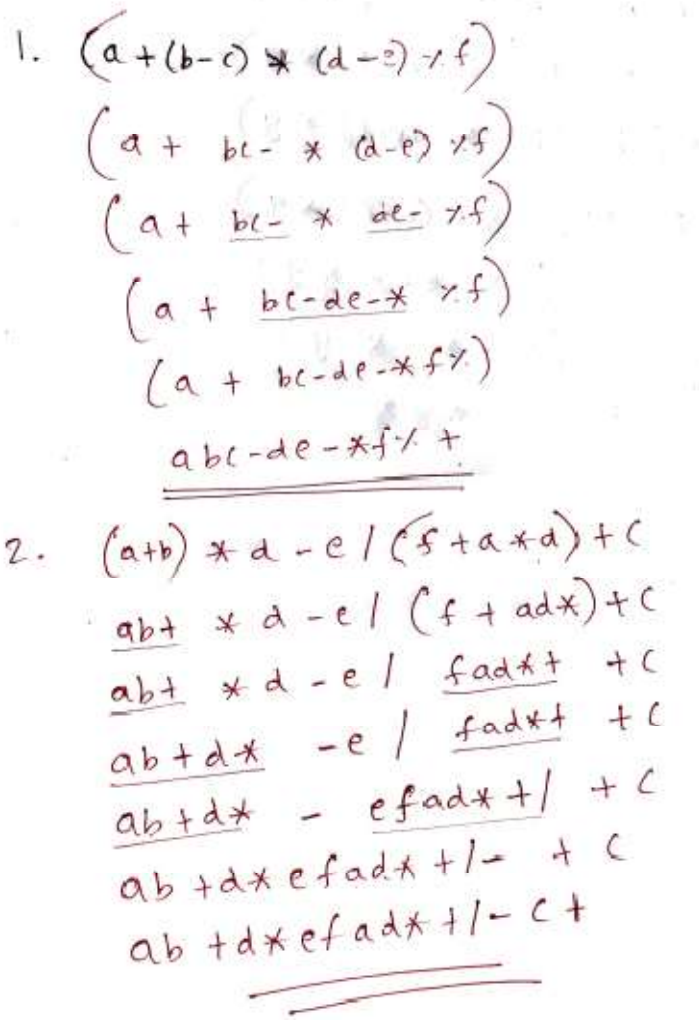
```
#include<stdio.h>
int a[20], n,i,pos,ele;
// Creating an Array of N Integer Elements
void create()
{
printf("\nEnter the number of elements\n");
scanf("%d",&n);
printf("\nEnter the elements one by one\n");
for(i=0;i<n;i++)
   scanf("%d",&a[i]);
}
//c. Display of Array Elements with Suitable Headings
void display()
{
printf("\n The array elements are:\n");
for(i=0;i<n;i++)
   printf("\na[%d]=%d",i,a[i]);
}
//a. Inserting an Element (ELEM) at a given valid Position (POS)
void insert()
{
printf("\nEnter the position where you wish to insert new element\n");
scanf("%d",&pos);
//check for valid position
if(pos>=n+1)
printf("Invalid position\n");
else
```

```
{
printf("\n Enter the element to be inserted:\t");
scanf("%d",&ele);
//shifting element to right
for(i=n-1;i>=pos;i--)
a[i+1]=a[i];
//inserts at specific position given
a[pos]=ele;
n++; //update number of elements
}
}
//b. Deleting an Element at a given valid Position(POS)
void delete()
{
printf("\nEnter the position from where you wish to delete element\n");
scanf("%d",&pos);
//check for valid position
if(pos>=n+1)
printf("Invalid position\n");
else
{
printf("\n Deleted Element is : %d",a[pos]);
//shifting element to left and delete the element form position
for(i=pos;i<n-1;i++)
   a[i]=a[i+1];
n--; //update number of elements
}
}
void main()
{
int choice;
// infinit loop for menu
while(1)// infinite loop for menu
{
printf("\n ARRAY OPERATIONS");
printf("\n Enter your choice");
printf("\n 1.CREATE\n 2.DISPLAY\n 3.INSERT \n 4.DELETE\n 5.EXIT\n");
scanf("%d",&choice);
switch(choice)
{
case 1:create();
break;
case 2:display();
break;
case 3:insert();
break;
case 4:delete();
break;
case 5:return;
```

```
default:printf("\n Invalid choice\n");
}
}
}
```

3a) Convert the following infix expressions to postfix expressions :
1. (a+(b-c)*(d-e)%f)
2. (a+b) * d – e / (f +a * d) +c
3. ((a / (b – c + d)) * (e – f) * g)

Solution:



3. ((a/(b-c+d))* (e-f) * g)
((a/(bc- +d)) * (e-f) * g)
((a / bc-d+) * (e-f) * g)
(abc-d+/ * (e-f) * g)
(abc-d+/ * ef- * g)
(abc-d+/ef-* * g)
abc-d+/ef-*g*

3b) Describe how to represent 1D and 2D arrays in memory with diagram.
Solution:

Representation of Linear Arrays in memory

- When the compiler encounters an array declaration such as **int list[5],** to create list, it allocates five consecutive memory locations. Each memory location is large enough to hold a single integer.
- The address of the first element list[0], is called the base address

base address=address(list[0])

- Using the base address the address of any element of list can be calculated using the formula**address(list[k])= base addres + w.k**

// where w is the size of each element in the array list

Example: **int list[5]**

- The elements of the array is list[0], list[1],…list[4]
- If the size of an integer on the machine is denoted by sizeof(int), then the memoryAddress(list [k])= base address+ sizeof (int).k.

Let α be the base address , the address of list[0] Let w=sizeof(int)
address(list[1])= α + w*1
address(list[2]) = α + w*2
address(list[3]) = α + w*3
address(list[4]) = α + w*4

Example : A 2 dimensional array A[3][4] is represented as

		0	1	2	3
	0		A[0][0]	A[0][1]	A[0][2]	A[0][3]
Rows	1		A[1][0]	A[1][1]	A[1][2]	A[1][3]
	2		A[2][0]	A[2][1]	A[2][2]	A[2][3]

Representation of two dimensional arrays in memory: A two dimensional m X n array A is storedin the memory as a collection of m*n sequential memory locations.If the array is stored column by column it is called column major order and if the array is stored row by row it is called row major order.

Example: Representation of the two dimensional array A[3][4] in row major order and column majororder

A	Subscript	
	A[0][0]	Row1
	A[0][1]	
	A[0][2]	
	A[0][3]	
	A[1][0]	Row2
	A[1][1]	
	A[1][2]	
	A[1][3]	
	A[2][0]	Row3
	A[2][1]	
	A[2][2]	
	A[2][3]	

Row Major Order

A	Subscript	
	A[0][0]	Column1
	A[1][0]	
	A[2][0]	
	A[0][1]	Column2
	A[1][1]	
	A[2][1]	
	A[0][2]	Column3
	A[1][2]	
	A[2][2]	
	A[0][3]	Column4
	A[1][3]	
	A[2][3]	

Column major order

4a) Discuss the representation of multidimensional arrays with examples.
Solution:

Representation of Multidimensional Arrays: In C, multidimensional arrays are represented using the array-of-arrays representation The linear list is then stored in consecutive memory just as we storea one-dimensional array.

If an n-dimensional array a is declared as a[upper0][upper1] ⋯ [uppern-1]; then the number of elements in the array is: $upper_0 * upper_1 * \ldots\ldots upper_{n-1}$ also represented as

$$\prod_{i=0}^{n-1} upper_i$$

where Π is the product of the $upper_i$'s. For instance, if we declare a as a[10][10][10], then we require $10 \cdot 10 \cdot 10 = 1000$ memory cell to hold the array. There are two common ways to represent multidimensional arrays: row major order and column major order. We consider only row major order here. As its name implies, row major order stores multidimensional arrays by rows.

Representation of an n-dimensional array

- Generalizing on the preceding discussion, we can obtain the addressing formula for anyelement A[i0][i1] … [in-1] in an n dimensional array declared as:
  A[upper0][upper1] … [uppern-1]

- If α is the address for A[0][0] … [0] the base address Then
  Address of a[i0][0][0][0] … [0] = α + i0 upper1 upper2 … upper n-1
  // address of a[i0][i1][0] … [0] = α+i0 upper1 upper2 … uppern-1 + i1 upper2 upper3 …uppern-1
- Repeating in this way the address for A[i0][i1] … [in-1] is=

$$\begin{aligned}
&\alpha + i_0 upper_1 upper_2 \ldots upper_{n-1} \\
&\quad + i_1 upper_2 upper_3 \ldots upper_{n-1} \\
&\quad + i_2 upper_3 upper_4 \ldots upper_{n-1} \\
&\quad \vdots \\
&\quad + i_{n-2} upper_{n-1} \\
&\quad + i_{n-1}
\end{aligned}$$

$$= \alpha + \sum_{j=0}^{n-1} i_j a_j \text{ where: } \begin{cases} a_j = \prod_{k=j+1}^{n-1} upper_k & 0 \le j < n-1 \\ a_{n-1} = 1 \end{cases}$$

4b) Describe self referential structures with an appropriate example.
Solution:

- A self-referential structure is one in which one or more of its data member is a pointer to itself. They require dynamic memory allocation (malloc and free) to explicitly obtain and release memory.
- These are the structures in which one or more pointers point to the structure of the same type.

 A Self-Referential Structure means when a structure is referencing another structure of same type.

Example:

```
struct self {
int p;
struct self *ptr; //It's a pointer to the same structure.
} ;
```

 Each instance of the structure self will have two components, p and ptr. p is a single integer variable, while ptr is a pointer to a self structure.
 The value of ptr is either the address in memory of an instance of self or the null pointer.

Example:

```
strcut code{
int i;
char c;
strcut code *ptr;
};
void main(){
struct code var1;
struct code var2;
var1.i=65;
var1.c='A';
var1.ptr=NULL;
var2.i=69;
var2.c='B';
var2.ptr=NULL;
var1.ptr=&var2; //With this statement var1 can access var2 members.
printf("%d%c", var1.ptr->i,var1.ptr->c);
}
```

Output:

66 B

5. Discuss the concept of dynamic 2D arrays with example. Write a C program to create and display dynamic 2D array.

Solution:

By using the concept of pointers with dynamic memory management, we can realize 2D dynamic arrays that are useful for many advanced applications. Consider the array declaration:

```
int a[3][4];
```

The compiler allocates 12 memory locations each of sizeof(int) bytes. In program, we may use less number of memory locations making rest of the memory locations useless. If an application wants more than 12 items to be stored, it is not possible in this case because only 12 memory locations are reserved by the compiler. It is at this point where dynamic storage becomes useful. Here, we can allocate required number of bytes to be accessed during run time.

The program shows how to create a 2D dynamic array and print the contents of the same:

```
#include<stdlib.h>
```

```
void main()
{
int row,col,i,j;
printf("\n enter the size of the array:");
scanf("%d%d",&row,&col);
int *arr=(int *) malloc(row*col*sizeof(int));
printf("\nenter 2D-array elements\n");
for(i=0;i<row;i++)
{
for(j=0;j<col;j++)
{
scanf("%d", (arr+i*col+j));
}
}
printf("\n2D-array elements are: \n");
for(i=0;i<row;i++)
{
for(j=0;j<col;j++)
{
printf("%d ", *(arr+i*col+j));
}
printf("\n");
}
free(arr);
}
```

6 a) What is realloc()? Explain with proper syntax and example program.

Solution:

Reallocation of memory(realloc): The function realloc resizes the memory previously allocated by either malloc or calloc.

syntax: Void * realloc (void * ptr , size_t new_size);

Example:

```
int *p;
p=(int*)calloc(n,sizeof(int))
p=realloc(p,s) /*where s is the new size*/
```

The statement realloc(p,s) -- Changes the size of the memory pointed by p to s. The existing contents of the block remain unchanged.

- When realloc is able to do the resizing it returns a pointer to the start of the new block
- When is not able to do the resizing the old block is unchanged and the function returns the value NULL.

Example program:

```
#include<stdio.h>
//To use realloc in our program
#include<stdlib.h>
```

```c
int main()
{
    int *ptr;

    //allocating memory for 10 integers
    ptr = malloc(10 * sizeof(int));

    //realloc memory size to store only 5 integers
    ptr = realloc(ptr, 5 * sizeof(int));

    return 0;
}
```

6b) List and explain any 3 applications of stack data structure.
Solution:
Application of stack
- Conversion of Expression
- Evaluation of expression
- Recursion

1. Conversion of Expression:
Compiler can not easily distinguish between the paranthesis and the operators. So we convert the expressions from infix to postfix or prefix form.
2. Evaluation of Postfix expression
In high level languages, infix notation cannot be used to evaluate expressions. Instead compilers typically use a parenthesis free notation to evaluate the expression. A common technique is to convert a infix notation into postfix notation, then evaluating it.
3.Recursion
A Function calls itself is called recursion where stack is used to store the status of each function call.

****