Q.2 a. Explain the difference between Type Conversion and Typecasting with the help of examples.

Answer:

Key: Refer Text Book1 , chapter 2 , page 25 & 27

Explanation: Type Conversion is that which converts to data type into another for example converting a int into float converting a float into double The Type Conversion is that which automatically converts the one data type into another but remember we can store a large data type into the other for example we can't store a float into int because a float is greater than int. When a user can convert the one data type into then it is called as the type casting Remember the type Conversion is performed by the compiler but a casting is done by the user for example converting a float into int Always Remember that when we use the Type Conversion then it is called the promotion when we use the type casting means when we convert a large data type into another then it is called as the demotion when we use the type casting then we can loss some data.

b. Explain in brief the following:

(i) Ternary Operator
(ii) Modulus Operator

Answer:

(i) Ternary Operator

Key: Refer Text Book1 , chapter 3 , page 37

Explanation: It is also known as the conditional operator (?:) is just like an if .. else statement that can be written within expressions.

The syntax of the conditional operator is

exp1 ? exp2 : exp3

- Here, exp1 is evaluated first. If it is true then exp2 is evaluated and becomes the result of the expression, otherwise exp3 is evaluated and becomes the result of the expression. For example,

  large = (a > b) ? a : b

  Conditional operators make the program code more compact, more readable, and safer to use

(ii) Modulus Operator

Key: Refer Text Book1 , chapter 3 , page 32

Explanation: Modulus Operator gives remainder of after an integer division, Assume variable A holds 10 and variable B holds 20 then: B % A will give 0

c. Write a program to find the maximum among five given numbers.

Answer:

Key: Text Book1 , chapter 1,2,3 , page 1-40 for basic constructs

Explanation: C program to Find max number from given numbers

```
#include<stdio.h>
#include<conio.h>
void main()
    {
     int lim,num,i,la;
     clrscr();
     printf("How many number you will type");
     scanf("%d",&lim);
     printf("Enter the numbers");
     for(i=1,la=0;i<=lim;i++)
     {
    scanf("%d",&num);
    if(num>la)      {
      la=num;
    }
     }
     printf("The maximum number among them is %d",la);
     getch();
    }
```

Q.3 a. Differentiate between the following concepts in C language:

(i) break and continue statement
(ii) while and do while constructs
(iii) switch statement and if-else statement

Answer: (i) break and continue statement

Key: Text Book1 , chapter 4 , page 54-55

Explanation: The break statement is used to terminate the execution of the nearest enclosing loop in which it appears.

• When compiler encounters a break statement, the control passes to the statement that follows the loop in which the break statement appears. Its syntax is quite simple, just type keyword break followed with a semi-colon.

break;

• In switch statement if the break statement is missing then every case from the matched case label to the end of the switch, including the default, is executed.

The continue statement can only appear in the body of a loop.When the compiler encounters a continue statement then the rest of the statements in the loop are skipped and the control is unconditionally transferred to the loop-continuation portion of the nearest enclosing loop. Its syntax is keyword followed with a semi-colon.

continue;

• If placed within a for loop, the continue statement causes a branch to the code that updates the loop variable. E.g

```
int i;
for(i=0; i<= 10; i++)
{ if (i==5)
continue;
printf("\t %d", i);
}
```

(ii) while and do while constructs

Explanation: Text Book1 , chapter 4 , page 51-52

WHILE LOOP

• The while loop is used to repeat one or more statements while a particular condition is true.

• In the while loop, the condition is tested before any of the statements in the statement block is executed.

• If the condition is true, only then the statements will be executed otherwise the control will jump to the immediate statement outside the while loop block.

• We must constantly update the condition of the while loop.

DO-WHILE LOOP

- The do-while loop is similar to the while loop. The only difference is that in a do-while loop, the test condition is tested at the end of the loop.
- The body of the loop gets executed at least one time (even if the condition is false).
- The do while loop continues to execute whilst a condition is true. There is no choice whether to execute the loop or not. Hence, entry in the loop is automatic there is only a choice to continue it further or not.
- The major disadvantage of using a do while loop is that it always executes at least once, so even if the user enters some invalid data, the loop will execute.
- Do-while loops are widely used to print a list of options for a menu driven program.

(iii) switch statement and if-else statement

Key: Text Book1 , chapter 4 , page 47-51

Explanation:

IF ELSE STATEMENT

• In the if-else construct, first the test expression is evaluated. If the expression is true, statement block 1 is executed and statement block 2 is skipped. Otherwise, if the expression is false, statement block 2 is executed and statement block 1 is ignored. In any case after the statement block 1 or 2 gets executed the control will pass to statement x. Therefore, statement x is executed in every case.

<u>SYNTAX OF IF STATEMENT</u>

```
if (test expression)
{
        statement_block 1;
}
else
{
        statement_block 2;
}
```

SWITCH STATEMENT
statement x;
A switch case statement is a multi-way decision statement. Switch statements are used:
When there is only one variable to evaluate in the expression
When many conditions are being tested for
• Switch case statement advantages include:
Easy to debug, read, understand and maintain
Execute faster than its equivalent if-else construct
General form :

```
switch(expression)
{
case constant:
statement;
break;
case constant:
statement;
break;
case constant:
statement;
break;
default:
statement;
break;
}
```

b. Write a program that reads a number and a single digit. It determines whether the number contains the digit or not.

Answer:

Key: Text Book1 , chapter 1-5 , page 1-50 for basic constructs
Explanation: void main()

```
{

// get the number to check
long int n;
intd;
printf("ENTER A NUMBER: ");
scanf("%ld",&n);
long int n1,mod;
printf("ENTER A digit: ");
scanf("%d",&d);


// n1=n;

// store the reversal in rev

while(n>0)
{
// grab the rightmost digit of n
mod = n%10;
if(d== mod)
printf("digit exists in number ");

// shift all digits of n to the right (removing the rightmost one)
n = n / 10;

}
}
```

Q.4 a. Describe array of pointers. Illustrate with proper example.
Answer:

Key: Text Book1 , chapter 9 , page 97
Explanation:
In general, an array of pointers can be used to point to an array of data items with each element of the pointer array pointing to an element of the data array. Data items can be accessed either directly in the data array, or indirectly by dereferencing the elements of the pointer array. The advantage of a pointer array is that the pointers can be reordered in any manner without moving the data items. For example, the pointer array can be reordered so that the successive elements of the pointer array point to data items in sorted order without moving the data items. Reordering pointers is relatively fast compared to reordering large data items such as data records or strings.
An array of pointers can be declared as :
<type> *<name>[<number-of-elements>];
For example :
char *ptr[3];
The above line declares an array of three character pointers.

b. Write a program that dynamically allocates an array of integers. A list of integers is read from keyboard and stores in the array. The program determines the smallest in the list and prints its location in the list.

Answer:

Key: Text Book1 , chapter 9 , page 85-98 for basic constructs

Explanation: #include <stdlib.h>

```
 Void main()
{
int *xPtr,n, min=0;
printf("\nEnter the number of values");
scanf("%d",&n);
/* Allocate space for n int values */
xPtr = (int *) malloc(n*sizeof(int));
 for(int i=0;i<n;i++)
printf("Enter the values of integers");
scanf("%d",*(ptr+i));
for(i=0;i<=n;i++)
     {
        if(*(ptr+i)<min)      {
      min=*(ptr+i);
    }
     }
     printf("The minimum number among them is %d",min);
     getch();
    }
```

Q.5 a. What is the difference between strings and character array?

Answer:

Key: Text Book1 , chapter 14 , page 133
Explanation: In C, a string is an array of characters with a terminating 0, or null character.

There's special syntax for specifying a string, which is the conventional quotes. "Fred" creates a constant character array, 5 characters long, with a terminal null. We call that a "strign literal". char name[5] = {'F', 'r', 'e', 'd', '\0'}; does exactly the same thing, but the array is writeable.

Occasionally you might need a character array without a terminating null. This wouldn't be a string in C terms, and will probably cause a crash if you feed it to a string function like strcpy(). Usually it's better to put on the null even if you don't use it, because it's only one byte, and it means you can print the string out with no trouble if you need to do so for some reason.

b. Define union with examples.

Answer:

Key: Text Book1 , chapter 16 , page 145

Explanation: The primary usefulness of a union is to conserve space, since it provides a way of letting many different types be stored in the same space. Unions also provide crude polymorphism. However, there is no checking of types, so it is up to the programmer to be sure that the proper fields are accessed in different contexts. The relevant field of a union variable is typically determined by the state of other variables, possibly in an enclosing struct.The size of a union is sufficient to contain the largest of its members. The value of at most one of the members can be stored in a union object at any time. A pointer to a union object, suitably converted, points to each of its members (or if a member is a bit-field, then to the unit in which it resides), and vice versa.

```
union <name>
{
    <datatype>  <1st variable name>;
    <datatype>  <2nd variable name>;
    .
    .
    .
    <datatype>  <nth variable name>;
} <union variable name>;
```

c. What is a File pointer? Write a program to copy the contents of one file into another.

Answer:

Key: Text Book1 , chapter 17 , page 147-150

Explanation: C communicates with files using a new datatype called a file pointer. This type is defined within stdio.h, and written as FILE *. A file pointer called output_file is declared in a statement like

```
  FILE *output_file;
```

```
#include<stdio.h>
int main(){
  FILE *p,*q;
  char file1[20],file2[20];
  char ch;
  printf("\nEnter the source file name to be copied:");
  gets(file1);
  p=fopen(file1,"r");
  if(p==NULL){
      printf("cannot open %s",file1);
      exit(0);
  }
  printf("\nEnter the destination file name:");
  gets(file2);
  q=fopen(file2,"w");
  if(q==NULL){
    printf("cannot open %s",file2);
    exit(0);
 }
 while((ch=getc(p))!=EOF)
    putc(ch,q);
 printf("\nCOMPLETED");
 fclose(p);
 fclose(q);
return 0;
}
```

Q.6 a. Input a two dimensional array of order m × n. Write an algorithm which should output this Array and another two dimensional array of order (m+1) × (n+1), in which the elements of (m+1) row should be the sum of elements of m rows and element of (n+1) column should be the sum of n columns.

Answer:

Key: Text Book1 , chapter 18 , page 167-173

Explanation: Algorithm:
Input: A matrix A of size m x n

Output: B matrix of size (m+1) x (n+1)

<u>Steps:</u>

```
Step1.  B: A matrix of size (m+1) x (n+1).

Step2. Initialize i=0

Step3. i=0,j=0
          While(i<m and j<n)
          B[i][j]<-A[i][j]
          i<-i+1 and j<-j+1

Step4. i<-0
          While(i<m)
          B[i][n+1]<- $^{n}\sum_{j=0}$ A[i][j]
          i<-i+1

Step5. j<-0
          While(j<n+1)
          B[m+1][j]<- $^{m}\sum_{i=0}$ A[i][j]
```

b. A list is ordered from smaller to largest when a sort is called. Which sorting algorithm would take the shortest time to execute? Justify.

Answer:

Key: Text Book1 , chapter 18 , page 195
Explanation: Quicksort is a divide and conquer algorithm. Quicksort first divides a large list into two smaller sub-lists: the low elements and the high elements. Quicksort can then recursively sort the sub-lists.
The steps are:
1. Pick an element, called a pivot, from the list.
2. Reorder the list so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final position. This is called the partition operation.
3. Recursively apply the above steps to the sub-list of elements with smaller values and separately the sub-list of elements with greater values.

The best case of the recursion is lists of size zero or one, which never need to be sorted. Quicksort algotithm is best than any other algorithm.

Quicksort is the most popular sorting algorithm. It's virtue is that it sorts in-place (even though it's a recursive algorithm) and that it usually is very fast. On reason for its speed is that its inner loop is very short and can be optimized very well.

Quicksort, or partition-exchange sort, is a sorting algorithm developed by Tony Hoare that, on average, makes O(n log n) comparisons to sort n items. In the worst case, it makes O(n^2) comparisons, though this behavior is rare. Quicksort is often faster in practice than other O(n log n) algorithms. Additionally, quicksort's sequential and localized memory references work well with a cache. Quicksort is a comparison sort and, in efficient implementations, is not a stable sort. Quicksort can be implemented with an in-place partitioning algorithm, so the entire sort can be done with only O(log n) additional space used by the stack during the recursion.

Q.7 a. Compare the stack and queue data structure. What is the difference between FIFO and LIFO? Give any three applications of stacks and queues each in System programming.

Answer:

Key: Text Book1 , chapter 19, page 227,235,242,256
Explanation: Stack:- A stack is an ordered list in which insertion and deletion are made at one end called top. Unlike that of the array, the definition of the stack provides for the insertion and deletion of the items, so that a stack is dynamic ,constantly changing object.
Example:-
A stack of coins, A stack of disk etc. As the items in this type of data structure can be added or removed from the top, it means the last item to b added to the stack is the first item to be removed. There stacks are also called Last In First Out (LIFO) lists.

Queue:- The queue is an ordered collection of items from which items may be deleted at one end(called the front of the queue)and into which items may be inserted at the other end (called the rear of the queue).

Example:-
A queue of student, a queue of book etc. As the items in this type of data structure can be added at the end(called rear) and removed from the first element(called front). it means the first item to be added to the stack is the first item to be removed. These stacks are also called First In First Out (FIFO) lists.

Comparison between stack and queue:-

1. Stack keeps the element in LIFO order.
   Whereas Queue keeps the element in FIFO order.

2. Stack uses push and pop operation for insert and delete element.
   Whereas Queue uses front and rear for insert and delete element.

Applications of Stack:-

a) Reversing a string: It reverses the order of execution. It includes pushing the individual characters and when the complete line is pushed onto the stack, and then individual characters of the line are popped off. Because the last inserted character pushed on stack would be the first character to be popped off, the line obviously be reserved.
b) Stack Frames: When any procedures or functions called, a number of words – the stack frame – is pushed onto a program stack. When the procedure or function returns this frame of data is popped off the stack. The stack is the region of main memory which programs temporarily store data as they execute.
c) Calculation of Postfix expression: The postfix notation is a type of notation which is most suitable for a computer to calculate any expression (due to reversing characteristic), and is the universally accepted notation for designing arithmetic and logical unit (ALU) of the CPU (processor).

Applications of Queues:-

i. Round Robin Technique for processor scheduling is implemented using queues.
ii. It is used by operating system.
iii. It is used in various scheduling algorithms.
   E.g. Disk Scheduling

b. Write a C function to merge two sorted linked lists into new sorted list.
Answer:

Key: Text Book1 , chapter 18 , page 180-184
Explanation: //PROGRAM

```
#include<stdio.h>
```

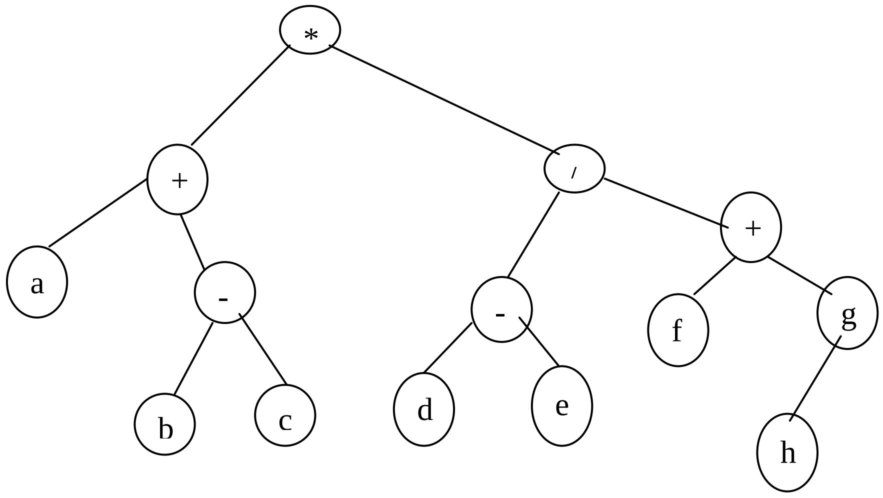
```
#include<stdlib.h>
typedefstruct Node
{
        int value;
        struct Node *next;
}node;

void merge(node * , node *);

void main()
{
        int i, n1=0,n2=0,val = 0;
        node *temp, *l1, *l2, *prev;
        clrscr();
        printf("Enter the no. of nodes in the first link list\n");
        scanf("%d", &n1);
        printf("Enter the values of the link list in order\n");
        scanf("%d", &val);
        l1 = (node *) malloc(sizeof(node));
        l1->value = val;
        l1->next = NULL;
        prev= l1;
        for(i=1;i<n1;i++)
        {
                scanf("%d", &val);
                temp = (node *) malloc(sizeof(node));
                temp->value = val;
                temp->next = NULL;
                prev->next = temp;
                prev = temp;
        }

printf("Enter the no. of nodes in the second link list\n");
        scanf("%d", &n2);
        printf("Enter the values of the link list in order\n");
        scanf("%d", &val);
        l2 = (node *) malloc(sizeof(node));
        l2->value = val;
        l2->next = NULL;
        prev = l2;
        for(i=1;i<n2;i++)
        {
                scanf("%d", &val);
                temp = (node *) malloc(sizeof(node));
                temp->value = val;
                temp->next = NULL;
                prev->next = temp;
                prev = temp;
        }
        merge(l1,l2);
        getch();
}
void merge(node * l1, node * l2)
{
        int i;
        node * head;
        node * prev, * temp;
        if(l1 == 0 || l2 == 0)
                return;
        if(l1->value < l2->value)
        {
                head = l1;
                l1 = l1->next;
        }
        else
        {
                head=l2;
                l2 = l2->next;
        }
        prev = head;

while(l1 != NULL && l2 != NULL)
        {
                if(l1->value < l2->value)
                {
                        prev->next = l1;
                        prev = l1;
                        l1 = l1->next;
                }
                else
                {
                        prev->next = l2;
                        prev = l2;
                        l2 = l2->next;
                }
        }
        if(l1==NULL)
                prev->next = l2;
        else
                prev->next = l1;

printf("The new sorted link list is:\n");
```

```
        temp = head;
        while(temp != NULL)
        {
                printf("%d ",temp->value);
                temp = temp->next;
        }
}
```

Q.8 a. Draw binary tree which denotes the following algebraic expression [a+(b-c)]×[(d-e)/(f+g-h)]
Change it into prefix expression using TRAVERSAL method.

Answer:

Key: Text Book1 , chapter 21 , page 354
Explanation:



b. Write an algorithm/ C program to delete a node from binary search tree.

Answer:

Key: Text Book1 , chapter 21 , page 375

```
Algorithm deleteBST (root, dltKey)
This algorithm deletes a node from a BST.
   Pre    root is reference to node to be deleted
          dltKey is key of node to be deleted
   Post   node deleted
          if dltKey not found, root unchanged
   Return true if node deleted, false if not found
1  if (empty tree)
   1  return false
2  end if
3  if (dltKey < root)
   1  return deleteBST (left subtree, dltKey)
4  else if (dltKey > root)
   1  return deleteBST (right subtree, dltKey)
5  else
      Delete node found--test for leaf node
   1  If (no left subtree)
      1  make right subtree the root
      2  return true
   2  else if (no right subtree)
      1  make left subtree the root
      2  return true
   3  else
         Node to be deleted not a leaf. Find largest node on
         left subtree.
      1  save root in deleteNode
      2  set largest to largestBST (left subtree)
      3  move data in largest to deleteNode
      4  return deleteBST (left subtree of deleteNode,
                           key of largest
   4  end if
6  end if
end deleteBST
```

Explanation:

Q.9 a. Define the following terms:

(i) Connected graph
(ii) Path
(iii) Directed Acyclic Graph

Answer:

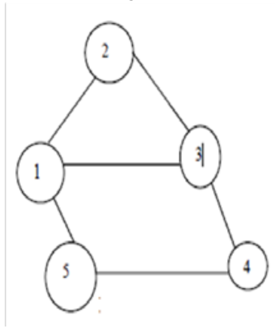
(i) Connected graph
Key: Text Book1 , chapter 22 , page 387-388
Explanation: Text Book1 , chapter 22 , page 387-388

Connected graph
An undirected graph is said to be **connected** iff for every pair of distinct vertices u and v in V(G) there is a path from u to v in G. Graph G1 is connected and graph G2 is not connected.

(ii) Path

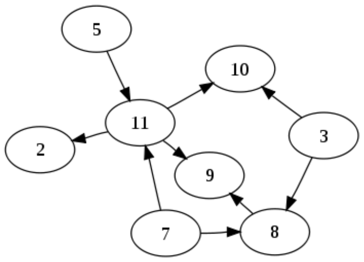
Key: Text Book1 , chapter 22 , page 387-388

- **Explanation:** Path:
  A path is defined as a sequence of distinct vertices, in which each vertex is adjacent to the next.
  A path, p, of length, k, through a graph is a sequence of connected vertices:
  $p = <v_0,v_1,...,v_k>$
- For example, the path from 1 to 4 can be defined as a sequence of adjacent vertices (1,5), (5,4).



(iii) Directed Acyclic Graph:

Explanation: is a directed graph with no directed cycles. That is, it is formed by a collection of vertices and directed edges, each edge connecting one vertex to another, such that there is no way to start at some vertex *v* and follow a sequence of edges that eventually loops back to *v* again



b. **What is topological sorting of a graph? Give an example to illustrate topological sort.**

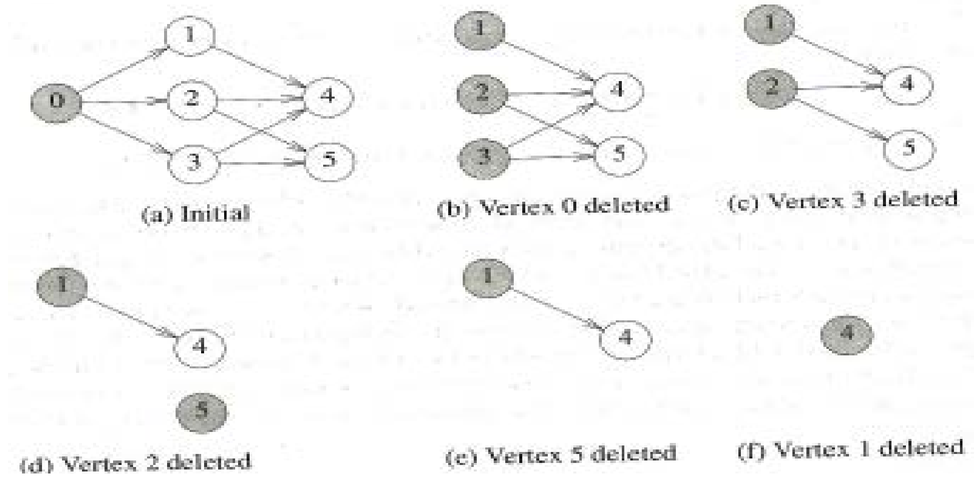
Answer:

Key: Text Book1 , chapter 22 , page 421-422

Explanation: Topological sorting of a graph is defined as a linear ordering of its vertices such that for every directed edge uv from vertex u to vertex v, u comes before v in the ordering. For instance, the vertices of the graph may represent tasks to be performed, and the edges may represent constraints that one task must be performed before another.

In case of graph, we make list of all the vertices first that have no predecessor. Then these vertex are deleted from the network along with all the edges leading out from it. Until all the vertices have been listed or all the remaining vertices in the network have predecessor and so none can be deleted.

Example: topological ordering for the following figure is-032514



TEXT BOOK

I. C & Data Structures, P.S. Deshpande and O.G. Kakde, Dreamtech Press, 2005.