DATA STRUCTURE CODING QUE.

Arrays by using c :

1). program to demonstrate arrays in c

```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
# define NUM_EMPLOYEE 10
int main (int argc, char *argv[])
{ int salary [NUM_EMPLOYEE], lcount=0,
    gcount=0 , i=0;
  printf ("Enter employee salary (MAX 10)\n");
     for (i=0; i<NUM_EMPLOYEE ; i++)
  {
    printf ("\n Enter employee salary : %d -",
          i+1) ;
    scanf ("%d ", & salary [i]);
  }
     for (i=0; i<NUM_EMPLOYEE ; i++)
  {  if (salary [i] <3000)
     lcount ++ ;
    else
     gcount ++;
  }
  printf ("\n There are {%d} employee with
        salary more than 3000 \n", gcount);
  printf ("There are {%d} employee with salary
        less than 3000 \n", lcount);
  printf ("press Enter to continue .. \n");
```

```
getchar ();
return 0;
}
```

2. Linked list in c++ :

```
using namespace std;
template <typename T>
class node
{
  public :
  T value;
  Node *next;
  Node *previous;
  Node (T value)
  {
    this->value = value;
  }
};
  template <typename T>
  class linked list
  {
    private :
    int size;
    Node<T> *head = NULL;
    Node<T> *tail = NULL;
    Node<T> *itr = NULL;
    public;
    Linked list()
    {
     this->size = 0;
    }
```

```
void append (T value)
{
 if (this->head == NULL)
{
 this->head - = new Node<T> (value);
 this->tail = this->head - ;
 }
else
 {
this - > tail_-> next = new node<T> (value)
this -> tail -> next - > previous = this_
                              ->tail ;

 }
this->size - + = 1 ;
 }
void prepend (T value)
 {
void resetIterator ()
 {
tail - = NULL ;
 }
int main (int argc, char **argv)
 {
    LinkedList<int> llist ;
    llist. append (10);
    llist. append (3);
    llist. append (1);
  cout<< "printing linked list <<endl;
  cout<<endl ;
  return 0;
 }
```

37. stack implementation in c :

```
#include <stdio.h>
int MAXSIZE = 8;
int stack [8];
int top = -1;
int isempty() {
   if (top == -1)
      return 1;
   else
      return 0;
   }
int isfull() {
   if (top == MAXSIZE)
      return 1;
   else
      return 0; }
int peek() {
   return stack [top]; }
int pop() {
   int data;
   if(!isempty()) {
      data = stack [top];
      top = top-1;
      return data; }
   else {
      printf("could not retriere data, stack is empty\n");
   }
}
int push (int data) {
   if (!isfull()) {
      top = top+1;
```

```
    stack[top] = data;
  } else {
    printf("could not insert data, stack is full\n");
  }
}
int main() {
  // push items on to the stack
  push(3);
  push(5);
  push(9);
  push(1);
  push(12);
  push(15);
  printf("Element at top of the stack : %d\n", peek());
  printf("Elements : \n");
  // print stack data
  while (!isempty()) {
    int data = pop();
    printf("%d\n", data);
  }
printf("stack full : %s\n", isfull() ? "true" : "false");
printf("stack empty : %s\n", isempty() "true", "false");
return 0;
}
```