Name - Ayush Jain
SAP ID - 60004200132
Div - B | B1

8-12-2021

Data Structure TT1

Q.1) Write a program to implement singly linked list using following functions : create, insert before, delete, display.

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

typedef struct node {
        int data;
        struct node *next;
           }node;

node* create_node(node *start, int val){
            node *ptr, *temp;
            ptr = (node*)malloc(sizeof(node));
            temp = start;
            Ptr -> data = val;
            Ptr -> next = NULL;
            if (start == NULL) {
                  Start = ptr;
               }
            else {
                  while (temp -> next != NULL) {
                        temp = temp -> next;
                        }
                  temp -> next = ptr;
               }
            return start;
       }
```

```
node *insert_before (node *start ,int val ,int bef) {
        node * ptr, *temp, *temp2 ;
        ptr = (node *) malloc (sizeof(node));
        temp = start ;
        temp2 = start ;
        Ptr -> data = val ;
        if (start == NULL)
        {
          Ptr -> next = NULL;
        Printf("There is no element in the list. So Element added
                to first position" );
        return ptr ;
       }
    else if (start -> data == bef) {
          ptr -> next = start ;
          return ptr;
        }
    else {
        while (temp ! = NULL && temp -> data ! = bef)
        {
          temp 2 = temp ;
          temp = temp -> next ;
        }
     temp 2 -> next = Ptr ;
     Ptr -> next = temp ;
     if ( temp = NULL) {
       printf(" %d not found in list.", bef );
       Printf("\n Hence element added to end of list");
       }
```

```
        return start ;
      }
  }
  void  display (node* start) {
          node* temp;
          temp = start ;
          if (start == NULL) {
              printf(" List is empty \n");
              return ;
          }
      Printf(" Nodes in singly linked list are : \n" );
      while ( temp -> next != NULL ) {
          Printf(" %d\n", temp -> data );
          temp = temp -> next ;
          }
      Printf (" %d\n", temp -> data);
  }

  struct node * delete_node (node* start)
      {
        int val;
        Printf("Enter the value of node to be deleted");
        scanf (" %d", &val);
        node* temp, * temp2, * todel;
        temp = start ;
        if (start == NULL) {
           printf (" List is empty");
        }
```

```
elseif (start -> data == val) {
        start = temp ->next;
        free (temp);
    }
  else {
     while (temp != NULL && temp-> data != val) {
           temp2 = temp;
           temp = temp -> next;
        }
       if (temp == NULL) {
          printf("%d is not in the list", val);
       } else {
             todel = temp;
             temp = temp -> next;
             temp -> next = temp;
             free (todel);
          }
       }
    return start;
}

int main () {
  node *start = NULL;
  int choice, val, num;
  do {
   printf(" \n Main Menu \n 1 to add a node \n 2 to insert a node
     before a value \n 3 to delete a node \n 4 to display the list \n
        5 to exit : ");
   scanf ("%d", &choice);
switch (choice) {
  case 1 :
     printf(" Enter a value : ") ;
     scanf ("%d ,&val);
     Start= create_node (start, val);
     break;
  case 2 :
    Printf(" Enter the number before which the link is to be inserted");
    scanf (" %d, &num) ;
    Printf(" Enter the values to be inserted ! ") ;
    scanf ("%d", & val);
    Start = insert_before (start ,val, num);
     break;
  case 3 :
    start = delete_node(start);
     break;
  case 4 :
     display (start);
     break;
  case 5 :
     break;
  default :
     printf (" wrong choice entered");
  }
} while (choice != 5);

}
```

Output:

```
Main Menu:
1 to add a node
2 to insert a node before a value
3 to delete a node
4 to display the list
5 to exit
 1
Enter a value: 5
 4
Nodes in singly linked list are : 5
 3
Enter the value of the node to be deleted : 5
```

Q. 2 write a Program to convert decimal number to binary using stack.

→

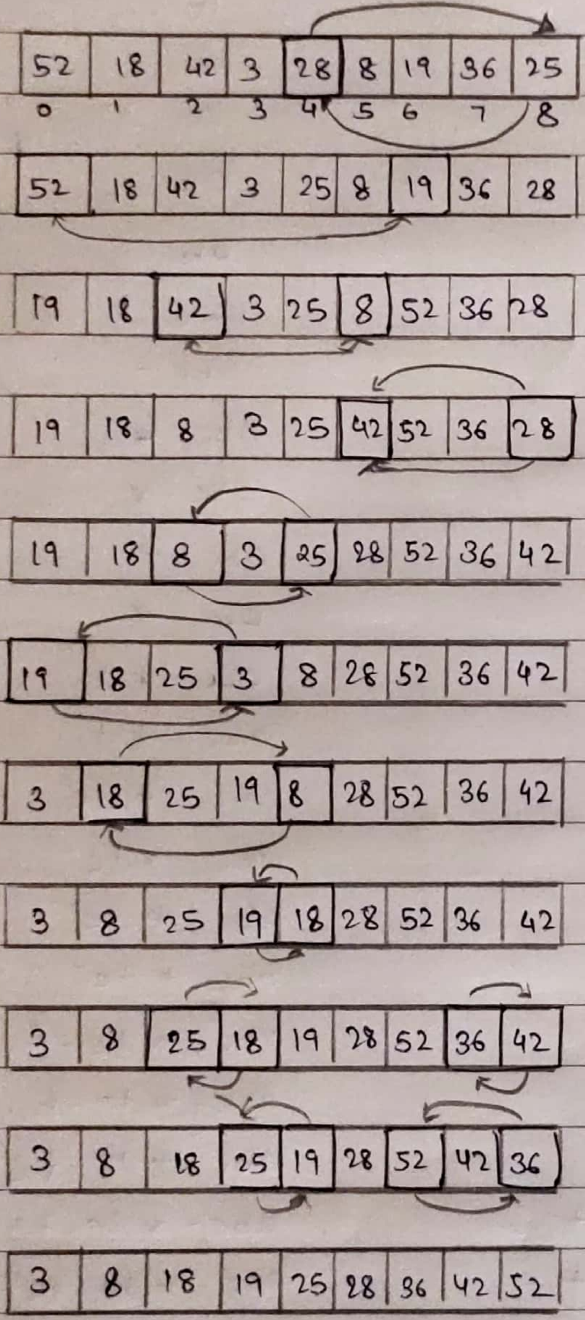
```
# include<stdio.h>
# include<stdlib.h>
# define MAX 100

int TOS = -1;
int stack[MAX];
void push (int value)
 {
     if(TOS = = MAX-1)
   Printf(" Stack overflow \n");
    else
     {
        TOS ++;
       stack [TOS]= value;
     }
}
int POP ()
{
  int value;
  if (TOS = = -1)
  {
    Printf (" Stack underflow \n");
     return -1;
  }
else
{
  value = stack [TOS];
   TOS -- ;
   return  value ;
 }
}
int  peek( )
{
     int  value;
     if (TOS == -1)
      {
        printf("Stack is empty \n");
        return -1;
      }
     else
     {
        value = stack [TOS];
         return value;
      }
}
void  convert(int n)
{
   while(n!=0)
    {
       push(n%2);
        n=n/2;
    }
}
```

```
void display ()
{
    int i;
    if (TOS = = -1)
     printf ("Stack is empty \n");
   else {
       for(i= TOS ; i>=0 ; i--)
        {
           printf (" %d,stack [i]);
        }
     }
}
int main() {
      int d;
      printf(" Enter the decimal number:");
      scanf (" %d", &d);
      convert (d);
      display();
      return 0;
}
```

<u>Output</u>:

```
Enter the decimal number: 7
 Binary number: 111
```

Q. 3) Demonstrate working of quick sort for following example
52, 18, 42, 3, 28, 8, 19, 36, 25



SORTED LIST

→ Select 28 as pivot

→ Move the left bound to the right bound till you reach the value greater that or equal to Pivot and interchange the values

→ When right bound crosses the left bound, all elements to the left of the bound are less than pivot and all elements to the right are greater. When that happens, interchange pivot with left bound.

→ Now call quick sort on left hand side of pivot and then right hand side.

→ Keep calling quick sort on left and right subset repeatedly till you get sorted list.

→ The principle behind quick sort is divide and conquer. It works by selecting a 'pivot' element from array and partitioning the other element into sub arrays. This is also called partition-exchange sort.

Q. 4)

Input	STACK	Output
(	(	
a	(	a
*	(*	a
(	(*(	a
b	(*(	ab
-	(*(-	ab
c	(*(-	abc
)	(*	abc-
)		abc-*
/	/	abc-*
(	/(	abc-*
(	/((	abc-*
d	/((	abc-*d
-	/((-	abc-*d
e	/((-	abc-*de
)	/(	abc-*de-
*	/(*	abc-*de-
(	/(*(	abc-*de-
f	/(*(	abc-*de-f
%	/(*(%	abc-*de-f
g	/(*(%	abc-*de-fg
+	/(*(~~%~~+	abc-*de-fg%
h	/(*(~~%~~+	abc-*de-fg%h
-	/(*(-	abc-*de-fg%h+
i	/(*(-	abc-*de-fg%h+i
)	/(*	abc-*de-fg%h+i-
)	/	abc-*de-fg%h+i-*
	Empty	abc-*de-fg%h+i-*/

Infix to post expression is abc-*de-fg%h+i-*/

60004200132

Q. 5) Differentiate between Array and Linked list.

Array	Linked list
1) An array is a colection of similar data types.	1) A linked list is collection of objects known as node where node consits of 2 points, i.e. data and address.
2) Array elements store in a continuous memory location.	2) Linked list element can be stored anywhere in memory.
3) Array works with a static memory. Here, static memory means that the memory size is fixed and cannot be changed during runtime.	3) Linked list works with dynamic memory. Here, dynamic memory means that the size can be changed at the runtime according to our requirements.
4) Array elements are independent of each other.	4) Linked list elements are connected as node contains the address of next node
5) Array takes more time while performing any operation like insertion, deletion, etc.	5) Linked list takes less time while performing operations like insertion, deletion, etc.
6) In case of an array, memory is allocated at compile-time.	6) In Linked list, memory is allocated at run-time.
7) Memory utilization is inefficient in the array.	7) Memory ~~ut~~ utilization is efficient in linked list