LAB ASSIGNMENT (WEEK 1- WEEK 3)

1. Implement STACK data structure using Array (Push,Pop, Display functions)

A. CODE:

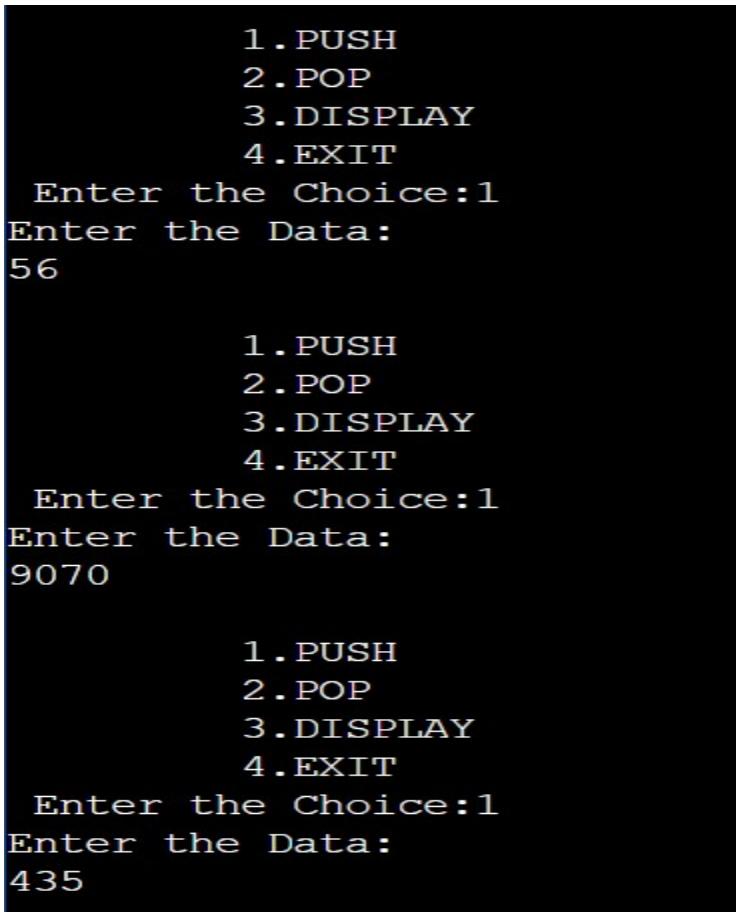
```
# include <stdio.h>
# define N 5
int stack[N];
int top=-1;
void push()
{
   int x;
   printf("Enter the Data: \n");
   scanf("%d",&x);
   if(top==N-1)
   {
      printf("Stack Overflow");
   }
   else
   {
      top++;
      stack[top]=x;
   }
}
void pop()
{
   int item;
   if(top==-1)
   {
```

```
        printf("Stack underflow");
    }
    else
    {
     item=stack[top];
     top--;
     printf("You popped item %d",item);
    }
}
void display()
{
    int i;
    for(i=top;i>=0;i--)
    {
        printf("%d \n",stack[i]);
    }
}
void main()
{
    int ch;
    do
    {
        printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
        printf("\n Enter the Choice:");
        scanf("%d",&ch);

        switch(ch)
        {
            case 1:
                push();
                break;
            case 2:
                pop();
```

```
            break;
         case 3:
         display();
            break;
         case 4:
           printf("\n\t EXIT POINT ");
            break;
         default:
            break;
      }
   }while(ch!=4);
}
```

SAMPLE INPUT AND SAMPLE OUTPUT:



```
          1.PUSH
          2.POP
          3.DISPLAY
          4.EXIT
 Enter the Choice:2
You popped item 435
          1.PUSH
          2.POP
          3.DISPLAY
          4.EXIT
 Enter the Choice:3
9070
56

          1.PUSH
          2.POP
          3.DISPLAY
          4.EXIT
 Enter the Choice:4

          EXIT POINT
```

2. Implement QUEUE data structure using Array (Enqueue, Dequeue, Display function)

A. CODE:

```
# include <stdio.h>
# define N 5
int queue[N];
int front=-1;
int rear=-1;
void enqueue()
{
   int x;
   printf("Enter the Data: \n");
   scanf("%d",&x);
   if(rear==N-1)
   {
      printf("Queue is Full or Overflow");
   }
   else if(front==-1 && rear==-1)
   {
      front=rear=0;
      queue[rear]=x;
   }
   else
   {
      rear++;
      queue[rear]=x;
   }
}
void dequeue()
{
```

```
    if(front==-1 && rear==-1)
    {
       printf("Queue is Empty or underflow");
    }
    else if(front==rear)
    {
       front=rear=-1;
    }
    else
    {
     printf("The dequeued Element is %d \n",queue[front]);
     front++;
    }
}
void display()
{
    int i;
    if(front==-1 && rear==-1)
    {
       printf("Queue ie Empty");
    }
    else
    {
    for(i=front;i<rear+1;i++)
    {
       printf("%d\n",queue[i]);
    }
    }
}
void main()
{
    int ch;
    do
```

```
    {
        printf("\n\t 1.Enqueue\n\t 2.Dequeue\n\t 3.DISPLAY\n\t 4.EXIT");
        printf("\n Enter the Choice:");
        scanf("%d",&ch);

        switch(ch)
        {
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("\n\tEXIT POINT");
                break;
            default:
                break;
        }
    }while(ch!=4);
}
```

SAMPLE INPUT AND SAMPLE OUTPUT:

```
          1.Enqueue
          2.Dequeue
          3.DISPLAY
          4.EXIT
 Enter the Choice:1
Enter the Data:
5498

          1.Enqueue
          2.Dequeue
          3.DISPLAY
          4.EXIT
 Enter the Choice:1
Enter the Data:
260

          1.Enqueue
          2.Dequeue
          3.DISPLAY
          4.EXIT
 Enter the Choice:1
Enter the Data:
841
```

```
          1.Enqueue
          2.Dequeue
          3.DISPLAY
          4.EXIT
 Enter the Choice:2
The dequeued Element is 5498

          1.Enqueue
          2.Dequeue
          3.DISPLAY
          4.EXIT
 Enter the Choice:3
260
841

          1.Enqueue
          2.Dequeue
          3.DISPLAY
          4.EXIT
 Enter the Choice:4

         EXIT POINT
```

3. Implement STACK data structure using Linked list (Push, Pop, Display functions)

A. CODE:

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int data;
   struct node *next;
};
struct node *top=0;
void push()
{
   struct node *newnode;
   newnode=(struct node *)malloc(sizeof(struct node));
   printf("Enter the data \n");
   scanf("%d",&newnode->data);
   newnode->next=top;
   top=newnode;
}
void display()
{
   struct node *temp;
   temp=top;
   if(top==0)
   {
      printf("STack is Empty");
   }
   else
   {
```

```
        while(temp!=0)
        {
            printf("%d \n",temp->data);
            temp=temp->next;
        }
    }
}
void pop()
{
    struct node *temp;
    temp=top;
    if(top==0)
    {
        printf("Stack is Empty");
    }
    else
    {
        printf("Popped Element is %d",top->data);
        top=top->next;
        free(temp);
    }
}
void main()
{
    int ch;
    do
    {
       printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
        printf("\n Enter the Choice:");
        scanf("%d",&ch);
    switch(ch)
    {
        case 1:
```

```
            push();
            break;
        case 2:
            pop();
            break;
        case 3:
            display();
            break;
        case 4:
             printf("\n\t EXIT POINT ");
            break;
        default:
            break;
    }
    }while(ch!=4);
}
```

SAMPLE INPUT AND SAMPLE OUTPUT:

```
          1.PUSH
          2.POP
          3.DISPLAY
          4.EXIT
 Enter the Choice:1
Enter the data
452

          1.PUSH
          2.POP
          3.DISPLAY
          4.EXIT
 Enter the Choice:1
Enter the data
7632

          1.PUSH
          2.POP
          3.DISPLAY
          4.EXIT
 Enter the Choice:1
Enter the data
89
```

```
          1.PUSH
          2.POP
          3.DISPLAY
          4.EXIT
 Enter the Choice:2
Popped Element is 89
          1.PUSH
          2.POP
          3.DISPLAY
          4.EXIT
 Enter the Choice:3
7632
452

          1.PUSH
          2.POP
          3.DISPLAY
          4.EXIT
 Enter the Choice:4

          EXIT POINT
```

4. Implement QUEUE data structure using Linked list (Enqueue, Dequeue, Display function)

A. CODE:

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int data;
   struct node *next;
};
struct node *front=0,*rear=0;
void enqueue()
{
   struct node *newnode;
   newnode=(struct node *)malloc(sizeof(struct node));
   printf("Enter the data \n");
   scanf("%d",&newnode->data);
   newnode->next=0;
   if(front==0 && rear==0)
   {
    front=rear=newnode;
   }
   else
   {
   rear->next=newnode;
   rear=newnode;
   }
}
void display()
{
```

```
    struct node *temp;

    if(front==0 && rear==0)
    {
        printf("Queue is Empty");
    }
    else
    {
        temp=front;
        while(temp!=0)
        {
            printf("%d\n",temp->data);
            temp=temp->next;
        }
    }
}
void dequeue()
{
    struct node *temp;
    temp=front;
    if(front==0 && rear==0)
    {
        printf("Queue is Empty");
    }
    else
    {
        printf("Dequeued Element is %d",front->data);
        front=front->next;
        free(temp);
    }
}
void main()
{
```

```
    int ch;
    do
    {
       printf("\n\t 1.Enqueue\n\t 2.Dequeue\n\t 3.DISPLAY\n\t 4.EXIT");
       printf("\n Enter the Choice:");
       scanf("%d",&ch);
    switch(ch)
    {
       case 1:
          enqueue();
          break;
       case 2:
          dequeue();
          break;
       case 3:
          display();
          break;
       case 4:
           printf("\n\tEXIT POINT");
          break;
       default:
          break;
    }
    }while(ch!=4);
}
```

SAMPLE INPUT AND SAMPLE OUTPUT:

```
          1.Enqueue
          2.Dequeue
          3.DISPLAY
          4.EXIT
 Enter the Choice:1
Enter the data
286

          1.Enqueue
          2.Dequeue
          3.DISPLAY
          4.EXIT
 Enter the Choice:1
Enter the data
498

          1.Enqueue
          2.Dequeue
          3.DISPLAY
          4.EXIT
 Enter the Choice:1
Enter the data
2109
```

```
          1.Enqueue
          2.Dequeue
          3.DISPLAY
          4.EXIT
 Enter the Choice:2
Dequeued Element is 286
          1.Enqueue
          2.Dequeue
          3.DISPLAY
          4.EXIT
 Enter the Choice:3
498
2109

          1.Enqueue
          2.Dequeue
          3.DISPLAY
          4.EXIT
 Enter the Choice:4

         EXIT POINT
```

5. Implement a singly linked list with the operations such as (Insert at the beginning, Insert at last, Insert after given value, delete the first node, delete the last node, delete the node with a given value).

A. CODE:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   int data;
   struct node *next;
};
struct node *head,*new_node,*temp,*prev_node,*next_node;
void main()
{
   create_list();
   insert_at_begin();
   insert_at_end();
   insert_at_position();
   delete_at_begin();
   delete_at_end();
   delete_at_position();
   display();
}
void create_list()
{
   int choice = 1;
   head = 0;
   while(choice == 1)
```

```
    {
        new_node = (struct node *)malloc(sizeof(struct node));
        printf("Enter the data : \n");
        scanf("%d",&new_node->data);
        new_node -> next = 0;
        if(head == 0)
        {
            head = temp = new_node;
        }
        else{
            temp -> next = new_node;
            temp = new_node;
        }
        printf("Do you want to continue ? if yes : Enter 1 no : Enter 0 \n");
        scanf("%d",&choice);
    }
}
void display()
{
    temp = head;
    printf("The remaing values in the nodes are : ");
    while(temp != 0)
    {
        printf("%d,",temp -> data);
        temp = temp -> next;

    }
}
void insert_at_begin()
{
    new_node = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of node to insert at begining : ");
    scanf("%d",&new_node -> data);
```

```
    new_node -> next = head;
    head = new_node;
    printf("Node inserted at the begining\n");
}
void insert_at_end()
{
    new_node = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of node to insert at end : ");
    scanf("%d",&new_node -> data);
    new_node -> next = 0;
    printf("Node inserted at the end \n");
    temp = head;
    while(temp -> next != 0)
    {
        temp = temp -> next;
    }
    temp -> next = new_node;
}
void insert_at_position()
{
    int pos, i = 1;
    new_node = (struct node *)malloc(sizeof(struct node));
    printf("enter the position of the node to insert: \n");
    scanf("%d",&pos);
    temp = head;
    while(i<pos)
    {
        temp = temp -> next;
        i++;
    }
    printf("Enter the value of node : ");
    scanf("%d",&new_node -> data);
    new_node -> next = temp -> next;
```

```
    temp -> next = new_node;
}
void delete_at_begin()
{
  temp = head;
  head = head -> next;
  free(temp);
  printf("Firt node deleted \n");
}
void delete_at_end()
{
  temp = head;
  while(temp -> next != 0)
  {
     prev_node = temp;
     temp = temp -> next;
  }
  if(temp == head)
  {
     head = 0;
     free(temp);
  }
  else
  {
     prev_node -> next = 0;
     free(temp);
  }
  printf("Last node deleted \n");
}
void delete_at_position()
{
  struct node *new_node;
  int pos, i = 1;
```

```
   printf("Enter the position of the node to delete : ");
   scanf("%d",&pos);
   temp = head;
   while(i<pos - 1)
   {
      temp = temp -> next;
      i++;
   }
    next_node = temp -> next;
    temp -> next = next_node -> next;
    free(next_node);
    printf("Node at given position is deleted \n");
}
```

SAMPLE INPUT AND SAMPLE OUTPUT:

```
Enter the data :
34
Do you want to continue ? if yes : Enter 1 no : Enter 0
1
Enter the data :
67
Do you want to continue ? if yes : Enter 1 no : Enter 0
1
Enter the data :
45
Do you want to continue ? if yes : Enter 1 no : Enter 0
0
Enter the value of node to insert at begining : 54
Node inserted at the begining
Enter the value of node to insert at end : 890
Node inserted at the end
enter the position of the node to insert:
2
Enter the value of node : 65
Firt node deleted
Last node deleted
Enter the position of the node to delete : 2
Node at given position is deleted
The remaing values in the nodes are : 34,67,45,
```

6. Implement two different programs for HASH table with Linear probing and chaining

A. CODE:

(i) HASH table by using linear probing

```
#include <stdio.h>
#include<stdlib.h>
#define TABLE_SIZE 4
int h[TABLE_SIZE]={NULL};
void insert()
{
 int key,index,i,flag=0,hkey;
 printf("\nenter a value to insert into hash table\n");
 scanf("%d",&key);
 hkey=key%TABLE_SIZE;
 for(i=0;i<TABLE_SIZE;i++)
    {
     index=(hkey+i)%TABLE_SIZE;
     if(h[index] == NULL)
     {
        h[index]=key;
         break;
     }
    }
    if(i == TABLE_SIZE)
     printf("\nelement cannot be inserted\n");
}
void search()
{
 int key,index,i,flag=0,hkey;
 printf("\nenter search element\n");
```

```
 scanf("%d",&key);
 hkey=key%TABLE_SIZE;
 for(i=0;i<TABLE_SIZE; i++)
 {
   index=(hkey+i)%TABLE_SIZE;
   if(h[index]==key)
   {
     printf("value is found at index %d",index);
     break;
   }
 }
 if(i == TABLE_SIZE)
   printf("\n value is not found\n");
}
void display()
{
 int i;
 printf("\nelements in the hash table are \n");
 for(i=0;i< TABLE_SIZE; i++)
 printf("\nat index %d \t value =  %d",i,h[i]);
}
void main()
{
   int opt,i;
   while(1)
   {
      printf("\n\t1.Insert\n\t2.Display\n\t3.Search\n\t4.Exit\t\n");
      printf("Enter a valid number : ");
      scanf("%d",&opt);

      switch(opt)
      {
         case 1:
```

```
            insert();
            break;
        case 2:
            display();
            break;
        case 3:
            search();
            break;
        case 4:
            exit(0);
      }
   }
}
```

(ii) HASH table by using chaining

```
#include <stdio.h>
#include <stdlib.h>
#define TABLE_SIZE 3
struct node
{
  int data;
  struct node *next;
};
struct node *head[TABLE_SIZE]={NULL},*c;
void insert()
{
    int i,key;
    printf("\nEnter a value to insert into hash table\n");
    scanf("%d",&key);
    i=key%TABLE_SIZE;
    struct node * newnode=(struct node *)malloc(sizeof(struct node));
    newnode->data=key;
```

```
    newnode->next = NULL;
    if(head[i] == NULL)
       head[i] = newnode;
    else
    {
       c=head[i];
       while(c->next != NULL)
       {
         c=c->next;
       }
       c->next=newnode;
    }
}
void search()
{
    int key,index;
    printf("Enter the element to be searched : ");
    scanf("%d",&key);
    index=key%TABLE_SIZE;
    if(head[index] == NULL)
       printf("Search element not found");
    else
    {
       for(c=head[index];c!=NULL;c=c->next)
       {
          if(c->data == key)
             {
                printf("search element found\n");
                break;
             }
       }
       if(c==NULL)
          printf("Search element not found\n");
```

```
    }
}
void display()
{
    int i;
    printf("\nelements in the hash table are \n");
     for(i=0;i<TABLE_SIZE;i++)
         {
             if(head[i] == NULL)
             {
             printf("No Hash Entry");

             }
             else
             {
               for(c=head[i];c!=NULL;c=c->next)
               printf("value =  %d\n",c->data);
             }
         }

}
main()
{
    int opt,key,i;
    while(1)
    {
        printf("\n\t1.Insert\n\t2.Display\n\t3.Search\n\t4.Exit\t\n");
        printf("Enter a valid number : ");
        scanf("%d",&opt);
        switch(opt)
        {
           case 1:
```

```
            insert();
            break;
         case 2:
            display();
            break;
         case 3:
            search();
            break;
         case 4:exit(0);

      }
   }
}
```

SAMPLE INPUT AND SAMPLE OUTPUT:

(i)

```
        1.Insert
        2.Display
        3.Search
        4.Exit
Enter a valid number : 1

enter a value to insert into hash table
4356

        1.Insert
        2.Display
        3.Search
        4.Exit
Enter a valid number : 1

enter a value to insert into hash table
879

        1.Insert
        2.Display
        3.Search
        4.Exit
Enter a valid number : 1
```

```
enter a value to insert into hash table
5

        1.Insert
        2.Display
        3.Search
        4.Exit
Enter a valid number : 2

elements in the hash table are

at index 0        value =  4356
at index 1        value =  5
at index 2        value =  0
at index 3        value =  879
        1.Insert
        2.Display
        3.Search
        4.Exit
Enter a valid number : 3

enter search element
5
value is found at index 1
```

```
        1.Insert
        2.Display
        3.Search
        4.Exit
Enter a valid number : 4
```

(ii)

```
        1.Insert
        2.Display
        3.Search
        4.Exit
Enter a valid number : 1

Enter a value to insert into hash table
57

        1.Insert
        2.Display
        3.Search
        4.Exit
Enter a valid number : 1

Enter a value to insert into hash table
676

        1.Insert
        2.Display
        3.Search
        4.Exit
Enter a valid number : 1
```

```
Enter a value to insert into hash table
67

        1.Insert
        2.Display
        3.Search
        4.Exit
Enter a valid number : 2

elements in the hash table are
value =  57
value =  676
value =  67
No Hash Entry
        1.Insert
        2.Display
        3.Search
        4.Exit
Enter a valid number : 3
Enter the element to be searched : 67
search element found
```

```
        1.Insert
        2.Display
        3.Search
        4.Exit
Enter a valid number : 4
```