G. Harsha vardhan
CSE-C5
AP19110010554

1) Construct a new linked list by merging alternate nodes of two lists.

Code:

```
#include <stdio.h>
#include <stdlib.h>
Struct Node
{
   int elements;
   Struct Node* next;
};
void PrintList(Struct Node* head)
{
   Struct Node* Ptr=head;
   while (Ptr)
   {
      Printf ("%d →", Ptr→elements);
      Ptr= Ptr→next;
   }
   printf ("Null \n");
void Push (Struct Node** head, int elements)
{
   Strut Node* NewNode = (Struct Node*)mallor (sizeof
                                    (Struct Node));
   NewNode→elements= elements;
   NewNode→next = *head
   *head =NewNode;
}
Struct Node* ShuffleMerge(Struct Node*a, Strut Node* b)
{  Strut Node link;
```

```
struct Node* tail = &link;
link.next = Null;
while(1)
{
    if (a == Null)
    {
        tail -> next = b;
        break;
    }
    else if(b=Null)
    {
        tail -> next = a;
        break;
    }
    else
    {
        tail -> next = a
        tail = a;
        a = a -> next;
        tail -> next = b;
        tail = b;
        b = b -> next;
    }
}
return link.next;
}
int main(void)
{
    int ele[] = {1,2,3,4,5,6,7,8,9,10,11};
    int n = sizeof(ele) / sizeof(ele[0]);
    struct Node* a = Null; *b = Null;
```

```
for (int i=n-1 ; i>=0 ; i=i-2)
{
    Push (&a, ele[i]);
}
for (int i=n-1 ; i>=0 ; i=i-2)
{
    Push (&b, ele[i]);
}
Printf (" first line :");
Printlist (a);
Printf (" second line :");
printlist (b);
Struct Node * head = Shuffle Merge (a, b);
 printf (" Merge:");
 printlist (head);
 return 0;
}
```

Output:

first line : 1 → 3 → 5 → 7 → 9 → 11 → Null

second line: 2 → 4 → 6 → 8 → 10 → Null

merge : 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 11 → Null.

2) i) array and linked list difference

Array	linked list
1) An array is a collection of element of a similar data type	1) linked lists is an ordered collection of element of same type in which each element is connected to next using pointers.
2) Array element can be accessed randomly using the array index	2) Random accessing is not possible in linked lists in elements will have to be accessed sequentially.
3) Data element are stored in contigous locations in memory	3) New elements can be stored anywhere and a refference is created for the new element using pointers.

2)ii) adding 1st node of linklist to other.

Code:

```
#include <stdio.h>
#include <stdlib.h>
Struct Node
{
   int element;
   Struct Node*next;
};
Void Printlist (Struct Node * head)
{
   Strut Node* ptr =head ;
```

```
while (ptr)
{
    Printf ("%d →" , Ptr → Element);
    Ptr = Ptr → next;
}
Printf ("Null \n");
}
Void Push (Struct Node ** head , int element )
{
  Struct Node * NewNode = (Struct Node *) malloc (sizeof
                                        (Struct Node));
  NewNode → Element = Element;
  NewNode → next = * head;
  * head = NewNode ;
}
Void moveNode (Strunt Node** destRef , Strut Node **
                                        SourceRef)
{
  if ( *SourceRef ==Null)
      return;
  Struct Node * NewNode = * SourceRef;
  * SourceRef = ( * SourceRef) → next;
  NewNode → next = * destRef;
  * destRef = NewNode;
}
int main (void)
{  int ele[] ={ 1, 2, 3}
   int n =sizeof (ele)/sizeof (ele[0]);
```

```
Struct Node *a = Null;
for (int i=n-1; i>=0; i--)
{
    Push(&a, ele(i));
}
Struct Node* b = Null;
for(int i=0; i<n; i++)
{
    Push(&b, 2* ele(i));
}
MoveNod(&a, &b);
Printf("first line:");
Printlist(a);
Printf("Second line:");
Print list(b);
return 0;
}
```

output:

first line: 6->1->2->3->null

Second line: 4->2->Null.

3)i) implement of queue in reverse order

Code:

```
#include <conio.h>
#include <stdio.h>
#define Max 100
Void Show(int stack[], int size, int top)
{
  int i;
  for( i=0; i<size; i++)
  {
    printf("\n value at %d IS %d", top, Stack(top));
    top = top - 1;
  }
}
Void reverse (int stack[], int q[], int *t, int*r,
                                        int*p)
{
  *p = 0;
  While (*t > -1)
  {
    *r = *r+1;
    q[*r] = Stack[*t];
    *t = *t -1;
  }
  While (*p <= *r)
  {
    *t = *t+1;
    Stack[*t] = q[*p];
    *p = *p+1
  }
```

```
int main()
{
    int size
    int element, t, i, stack[MAX], q[MAX];
    int top=-1, front=-1, rear=-1;
    printf("Enter size of stack");
    scanf("%d", &size);
    for(i=0; i<size; i++)
    {
        top=top+1;
        printf("enter the value of position %d:", top);
        scanf("%d", &element);
        stack(top)=element;
    }
    show(stack, size, top);
    reverse(stack, queue, &top, &rear, &Front);
    printf("\n After reverse");
    show(stack, size, top);
    getch();
}
```

Output:

```
Enter size of Stack : 3
Enter value of position 0 : 1
Enter value of position 1 : 2
Enter value of position 2 : 3
Value at 2 is 3
Value at 1 is 2
Value at 0 is 1
After reverse
Value at 2 is 1
Value at 1 is 2
Value at 0 is 3
```

---

3) ii) program to print elements in queue in alternate order.

```
# include <stdio.h>
# define Max 100
Void insert( );
void alternate( );
void display ( )
int qarray [MAX];
int r = -1;
int f = -1 , size;
```

```
Scanf ( "%d" , &Size);
iwMain()
{
    int choice;
    while(1)
    {
        Printf ("1. Insert element to queue \n");
        Printf ("2. Display element from queue \n");
        Printf ("3. Alternate element ");
        Printf ("4. Quit \n");
        Printf ("Enter choice");
        Scanf ("%d", &choice);
        Switch (choice)
        {
            case 1:
            {
                insert();
                break;
            }
            case 2:
            {
                display();
                break;
            }
            case 3:
            {
                alternate();
                break;
            }
```

```
        case 4:
        {
            exit(1);
        }
        default :
        {
            printf (" wrong choice \n");
        }
      }
    }
  }
void insert()
{
    int addelement ;
    if (r == Max-1)
    Printf (" Queue overflow \n" );
    else
    {
        if (f ==-1)
        f = 0;
        Printf ("Insert the element in Queue");
        Scanf (" %d", & addelement);
        r = r+1
        qarray [r] = addelement ;
    }
}
```

```
void display()
{
    int i;
    if (f == -1)
    {
        Printf("Queue is empty \n");
    }
    else
    {
        Printf("Queue is: \n");
        for(i = f; i <= r; i++)
        {
            Printf("%d", qarray[i]);
        }
        Printf("\n");
    }
}
Void alternate()
{
    int i, j, temp;
    Printf("Alternate Elements are \n");
    for(i = 0; i < size; i = i+2)
    {
        Printf("%d \n", qarray[i]);
    }
}
```

Output:

Enter choice : 1

Insert the element in queue : 100

Enter choice : 1

Insert the elemt in queue : 200

Enter choice : 1

Insest the element in queue : 300

Enter choice : 2

100
200
300

Enter choice : 3

100
300

Enter choice : 4

Exit.

---

4) all elements in stack whose sum is k

Code:

```
#include <stdio.h>
int Stack[100], choice, n, top, x, i;
void push(void);
void display(void);
```

```
int main( )
{
    top = -1;
    Printf ("\n enter the size of Stack : ");
    Printf (- Scanf ("%d", &n);
    Printf ("\n\t STACK OPERATIONS USING ARray");
    Printf ("\n\t 1. Push \n\t 2. Display \n\t 3.Subarray
                                    \n\t 4. EXIT");
    do
    {
      Printf ("\n enter choice :");
      Scanf ("%od , &choice);
      Switch (choice)
      {
        case 1:
        {
          Push()
          break;
        }
        case 2:
        {
          display()
          break;
        }
        case 3:
        {
          SubArray( )
          break;
        }
```

```
        case 4:
        {
            Printf("\n\t EXIT Point");
            break;
        }
        default:
        {
            printf("\n\t Enter choice from 1 to 4");
        }
    }
}
while (choice != 4)
return 0;
}
Void Push()
{
    if (top >= n-1)
    {
        Printf("\n\t stack overflow");
    }
    else
    {
        printf("enter value to Push");
        Scanf("%d", &x);
        top ++
        Stack (top) = x;
    }
}
```

```
Void display()
{
    if(top >= 0)
    {
        Printf("\n the elemends in STack \n");
        for (i = top ; i>=0; i--)
        {
            printf("\n %d", Stack[i]);
        }
        printf("\n Press next choise");
    }
    else
    {
        Printf("\n stack is empty");
    }
}
int Subarray sum (int Stack[ ], int sum).
{
    int currentsum ,i , j ;
    Scanf("%d", &sum);
    for(i=0; i<n ; i++)
    {
        currentsum = Stack[i] ; Stack[J];
        for (j =i+1 ; j<=n ; j++)
        {
            if (currentsum ==sum)
            {
                printf("sum found %d and %, Stack[i]
                                        Stack[j]);
                return 1;
            }
```

```
        If (currentSum > Sum || j==n)
            break;
        currentSum = currentSum + Stack[j];
        }
    }
    Printf ("no Subarray found");
    return 0;
}

int main()
{
    int Sum = 23;
    Sub arraySum (Stack, n, Sum);
    return 0;
}
```

output:

```
1. Push
2. Displays
3. Subarray
4. EXIT
Enter choice = 1
enter value to Push : 1
1
enter choice : 1
enter value to Push:
2
enter choice : 1
enter value to Push:
3
```

Enter choice : 2

Enter value

The elements in stack

1

2

3

Press next choice : 3

3

Sum found 1, 2

---

5) Program to insert and delete an element at n^{th} & k^{th} Position in linklist.

Code:

```
#include <stdio.h>
#include <stdlib.h>
Struct linklist
{
    int num;
    Struct linklist *next;
};
typedef Struct linklist node;
node *head = Null, *tail = Null;
Void Createlinklist();
Void Printlinklist();
Void insertlinklistatlast(int element);
Void insertlinklistatfirst(int element);
Void insertlinklistatafter(int value, int element);
Void delliteitem(int element);
Void searchitem(int element);
```

```
int main()
{
    int value, element;
    Printf("Create linklist \n");
    Createlinklist();
    Printlinklist();
    Printf("\n Insert new item at last \n");
    Printf("%d", &element);
    Insertlinklistatlast(element);
    Printlinklist();
    Printf("\n insert new item at first \n");
    Scanf("%d", &element);
    Insertlinklistatfirst(element);
    Printlinklist();
    Printf("\n enter a value (existing item of list),
                after that you want to insert
                                    a value \n");
    Scanf("%d", &value);
    printf("\n Insert new item after %d value \n", value);
    Scanf("%d", &element);
    Insertlinklistatafter(value, element);
    Printlinklist();
    Printf("\n Enter an item to search it from list \n");
    Scanf("%d", &element);
    Searchitem(element);
```

```
    Printf("\n enter an element, which your want to delete:\n");
    Scanf ("%d", element);
    deleteitem (element);
    Printlinklist ( );
    return 0;
}
Void createlinklist( )
{
    int ele;
    while (1)
    {
        Printf("Input a number.(enter -1 to exit) \n");
        scanf("%d", &element);
        if (ele == -1)
            break;
        insertlinklistatlast(ele);
    }
}
void insertlinklistatlast (int element)
{
    node *temp_node;
    temp-node = (node *) malloc( sizeof(node));
    temp-node → number = element
    temp_node → next = Null;
```

size of
(Structu

```
if (head = Null)
 {
    head = temp-node
    tail = temp-node;
 }
 else
 {
    tail → next = temp-node
    tail = temp-node;
 }
}

Void insertlinklistatfirst (int element)
{
   node * temp-node = (node *) malloc(sizeof(node));
   temp-node → num = element;
   temp-node → next = head;
   head = temp-node;
}

Void insertlintlistat after (int value, int element)
{
   node * myNode = head
   int flag = 0;
   while (my Node != Null)
   {
      if (myNode → number == value)
      {
         node * newnode = (node *) malloc (sizeof (node));
```

```
        newnode -> num = element;
        newNode -> next = myNode -> next;
        myNode -> next = newNode;
        Printf ("%d is inserted after %d \n", element,
                                                value);
        flag = 1;
        break;
    }
    else
        myNode = myNode -> next;
}
if (flag == 0)
  {
    Printf ("Value not found \n");
  }
}
Void deleteitem (int ~~value~~ element)
{
    node * myNode = head ; * previous = NULL;
    int flag = 0;
    while (myNode != Null)
    {   if (myNode -> number == element)
        {
            if (Previous == NULL)
            {
                head = myNode -> next
            }
```

```
else
{
    Previous->next = myNode -> next;
}
printf(" %d is deleted from list \n", elemnt);
flag = 1;
free (myNode);
break;
}
Previs = myNode;
myNode = myNode -> next;
}
if (flag == 0)
{
  printf (" value not found \n");
}
}
Void PrintLinklist ()
{
  Printf ("\n Yourfull linklist is \n");
  node *mylist;
  mylist = head
  while (mylist != Null)
  {
    Printf ("%d", mylist -> num);
    mylist = mylist -> next
  }
  puts (' ');
}
```

Output:

```
Create linklist
Input a number (enter -1 to exit)
1
Input a number (enter -1 to exit)
2
Input a number (enter -1 to exit)
3
Input a number (enter -1 to exit)
4
Input a number (enter -1 to exit)
5
Input a number (enter -1 to exit)
6
Input a number (enter -1 to exit)
-1
Your full linklist is
1 2 3 4 5 6
Insert new item at last
1 2 3 4 5 6 7
Insert new item at first
0
Your full linklist is
0 1 2 3 4 5 6 7
Enter a Value (existing item of list),
        after that you want to insert a value
7
```

Enter new item after 7 value

8

8 is inserted after 7

Your full linklist is

0 1 2 3 4 5 6 7 8

Enter an item to search it from list

3

3 is present in list. Memory address is 12348688

Enter a value, which you want to delete from liste

6

6 is deleted from list

Your full linked list is

0 1 2 3 4 5 7 8