Q1. Write a C program to create two sorted single linked list and then merge them.

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

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

struct node *insert(struct node *head, int x)
{
   struct node *nn = (struct node *)malloc(sizeof(struct node));
   nn->data = x;
   nn->next = NULL;

   if (head == NULL)
   {
      return nn;
   }
   struct node *curr = head;
   while (curr->next != NULL)
   {
      curr = curr->next;
   }
   curr->next = nn;
   return head;
}

void disp(struct node *head)
{
   struct node *curr = head;
   while (curr != NULL)
   {
      printf("%d ", curr->data);
      curr = curr->next;
   }
   printf("\n");
}

struct node *merge(struct node *l1, struct node *l2)
{
   if (l1 == NULL)
      return l2;
   if (l2 == NULL)
      return l1;
```

```
    struct node *NewNode = NULL;

    if (l1->data <= l2->data)
    {
        NewNode = l1;
        NewNode->next = merge(l1->next, l2);
    }
    else
    {
        NewNode = l2;
        NewNode->next = merge(l1, l2->next);
    }

    return NewNode;
}

int main()
{
    struct node *l1 = NULL;
    struct node *l2 = NULL;
    struct node *nn = NULL;

    int ch, x;

    printf("\n\nKrishant Chauhan MCA 2A Roll no-32 \n");
    printf("1. Insert into List 1\n");
    printf("2. Insert into List 2\n");
    printf("3. Merge Lists\n");
    printf("4. Display Merged List\n");
    do
    {
        printf("\nEnter your choice: ");
        scanf("%d", &ch);

        switch (ch)
        {
        case 1:
            printf("Enter the x to insert into List 1: ");
            scanf("%d", &x);
            l1 = insert(l1, x);
            break;
        case 2:
            printf("Enter the x to insert into List 2: ");
            scanf("%d", &x);
            l2 = insert(l2, x);
            break;
        case 3:
            nn = merge(l1, l2);
            printf("Lists merged successfully!\n");
            break;
        case 4:
            printf("Merged List: ");
            if (nn == NULL)
                printf("Please Merge them\n");
            disp(nn);
            break;
        default:
            printf("Invalid choice! Please try again.\n");
            break;
        }
    } while (ch != 5);
    return 0;
}
```

OUTPUT :-

```
Krishant Chauhan MCA 2A Roll no-32
1.Insert into List 1
2.Insert into List 2
3.Merge Lists
4.Display Merged List

Enter your choice: 1
Enter the x to insert into List 1: 10
Enter your choice: 1
Enter the x to insert into List 1: 20
Enter your choice: 1
Enter the x to insert into List 1: 30
Enter your choice: 1
Enter the x to insert into List 1: 40
Enter your choice: 2
Enter the x to insert into List 2: 100
Enter your choice: 2
Enter the x to insert into List 2: 200
Enter your choice: 2
Enter the x to insert into List 2: 300
Enter your choice: 3
Lists merged successfully!
Enter your choice: 4
Merged List: 10 20 30 40 100 200 300
```

Q2. Write a C program to create a single linked list ,then write another function InsertNth() which can insert a new node after any node given by the user in that linked list.

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

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

struct node *append(struct node *head, int x)
{
   struct node *nn = (struct node *)malloc(sizeof(struct node));
   nn->data = x;
   nn->next = NULL;

   if (head == NULL)
   {
      return nn;
   }
   struct node *curr = head;
   while (curr->next != NULL)
   {
      curr = curr->next;
   }
   curr->next = nn;
   return head;
}

void display(struct node *head)
{
   struct node *current = head;
   while (current != NULL)
   {
      printf("%d ", current->data);
      current = current->next;
   }
   printf("\n");
}

struct node *insert(struct node *head, int val, int pos)
{
   struct node *temp, *newnode;
   int i = 1;
   newnode = (struct node *)malloc(sizeof(struct node));
   newnode->data = val;
```

```
    if (pos == 1)
    {
        newnode->next = head;
        head = newnode;
    }
    else
    {
        temp = head;
        while (i < pos - 1 && temp != NULL)
        {
            temp = temp->next;
            i++;
        }
        if (temp == NULL)
        {
            printf("Invalid position\n");
            return head;
        }
        newnode->next = temp->next;
        temp->next = newnode;
    }
    printf("Value Inserted at %dth Position\n", pos);
    return head;
}

int main()
{
    struct node *head = NULL;
    int ch, x, pos;
    printf("\n\nKrishant Chauhan MCA 2A Roll no-32 \n");
    printf("1. Append\n");
    printf("2. Insert\n");
    printf("3. Display\n");
    printf("4. Exit\n");
    do
    {
        printf("Enter your choice: ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            printf("Enter the value to append: ");
            scanf("%d", &x);
            head = append(head, x);
            break;
        case 2:
            printf("Enter the value to insert: ");
            scanf("%d", &x);
            printf("Enter the position ");
            scanf("%d", &pos);
            head = insert(head, x, pos);
            break;
        case 3:
            printf("Linked List :-\n");
            display(head);
            break;
        case 4:
            printf("Exit\n");
            break;
        default:
            printf("Invalid choice!\n");
            break;
        }
    } while (ch != 4);

    return 0;
}
```

OUTPUT :-

```
Krishant Chauhan MCA 2A Roll no-32
Append
Insert
Display
Exit

Enter your choice: 1
Enter the value to append: 10
Enter your choice: 1
Enter the value to append: 20
Enter your choice: 1
Enter the value to append: 30
Enter your choice: 1
Enter the value to append: 40
Enter your choice: 1
Enter the value to append: 50
Enter your choice: 2
Enter the value to insert: 450
Enter the position 5
Value Inserted at 5th Position
Enter your choice: 3
Linked List :-
10 20 30 40 450 50
```

Q3. Write a C program to create single linked then remove duplicate nodes in unsorted linked list.

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

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

struct node *insert(struct node *head, int x)
{
   struct node *nn = (struct node *)malloc(sizeof(struct node));
   nn->data = x;
   nn->next = NULL;

   if (head == NULL)
   {
      return nn;
   }
   struct node *curr = head;
   while (curr->next != NULL)
   {
      curr = curr->next;
   }
   curr->next = nn;
   return head;
}
void disp(struct node *head)
{
   struct node *curr = head;
   while (curr != NULL)
   {
      printf("%d ", curr->data);
      curr = curr->next;
   }
   printf("\n");
}
void rem(struct node *head)
{
   struct node *current = head;
   struct node *runner;
   struct node *duplicate;

   while (current != NULL && current->next != NULL)
   {
      runner = current;
```

```
        while (runner->next != NULL)
        {
            if (current->data == runner->next->data)
            {
                duplicate = runner->next;
                runner->next = runner->next->next;
                free(duplicate);
            }
            else
                runner = runner->next;
        }
        current = current->next;
    }
}

int main()
{
    struct node *head = NULL;
    int choice, value;
    printf("\n\nKrishant Chauhan MCA 2A Roll no-32 \n");
    printf("1. Insert a node\n");
    printf("2. Remove duplicate nodes\n");
    printf("3. Display \n");
    printf("4. Exit\n");
    do
    {
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Enter the value: ");
            scanf("%d", &value);
            head = insert(head, value);
            break;
        case 2:
            rem(head);
            printf("Duplicate nodes removed \n");
            break;
        case 3:
            printf("List: ");
            disp(head);
            break;
        case 4:
            printf("Exit\n");
            break;
        default:
            printf("Invalid choice!\n");
            break;
        }
        printf("\n");
    } while (choice != 4);

    return 0;
}
```

OUTPUT :-

```
Krishant Chauhan MCA 2A Roll no-32

Insert a node
Remove duplicate nodes
Display
Exit
Enter your choice: 1
Enter the value: 10
Enter your choice: 1
Enter the value: 10
Enter your choice: 1
Enter the value: 20
Enter your choice: 1
Enter the value: 30
Enter your choice: 1
Enter the value: 40
Enter your choice: 3
List: 10 10 20 30 40
Enter your choice: 2
Duplicate nodes removed
Enter your choice: 3
List: 10 20 30 40
```

Q4. Write a C program to create a linked list P, then write a 'C' function named **split** to create two linked lists Q & R from P So that Q contains all elements in odd positions of P and R contains the remaining elements. Finally print both linked lists i.e. Q and R.

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

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

struct node *append(struct node *head, int x)
{
   struct node *nn = (struct node *)malloc(sizeof(struct node));
   nn->data = x;
   nn->next = NULL;

   if (head == NULL)
   {
      return nn;
   }
   struct node *curr = head;
   while (curr->next != NULL)
   {
      curr = curr->next;
   }
   curr->next = nn;
   return head;
}

void display(struct node *head)
{
   struct node *current = head;
   while (current != NULL)
   {
      printf("%d ", current->data);
      current = current->next;
   }
   printf("\n");
}

void insert(struct node *head, struct node **Q, struct node **R)
{
   struct node *curr = head;
   struct node *qhead = NULL;
   struct node *rhead = NULL;
```

```
int flag = 1;

while (curr != NULL)
{
   if (flag == 1)
   {
      if (*Q == NULL)
      {
         *Q = curr;
         qhead = *Q;
         curr = curr->next;
      }
      else
      {
         (*Q)->next = curr;
         (*Q) = (*Q)->next;
         curr = curr->next;
      }
      flag = 0;
   }
   else if (flag == 0)
   {
      if (*R == NULL)
      {
         *R = curr;
         rhead = *R;
         curr = curr->next;
      }
      else
      {
         (*R)->next = curr;
         (*R) = (*R)->next;
         curr = curr->next;
      }
      flag = 1;
   }
}

if (*Q != NULL)
{
   (*Q)->next = NULL;
}
if (*R != NULL)
{
   (*R)->next = NULL;
}

*Q = qhead;
```

```
    *R = rhead;
}

int main()
{
    struct node *head = NULL, *Q = NULL, *R = NULL;
    int ch, x, pos;
    printf("\n\nKrishant Chauhan MCA 2A Roll no-32 \n");
    printf("1. Append\n");
    printf("2. Split\n");
    printf("3. Display\n");
    printf("4. Exit\n");
    do
    {
        printf("Enter your choice: ");
        scanf("%d", &ch);

        switch (ch)
        {
        case 1:
            printf("Enter the value to append: ");
            scanf("%d", &x);
            head = append(head, x);
            break;
        case 2:
            insert(head, &Q, &R);
            break;
        case 3:
            printf("Odd Position\n");
            display(Q);
            printf("Even Position\n");
            display(R);
            break;
        case 4:
            printf("Exit\n");
            break;
        default:
            printf("Invalid choice!\n");
            break;
        }
    } while (ch != 4);

    return 0;
}
```

OUTPUT :-

```
Krishant Chauhan MCA 2A Roll no-32
Append
Split
Display
Exit
Enter your choice: 1
Enter the value to append: 10
Enter your choice: 1
Enter the value to append: 20
Enter your choice: 1
Enter the value to append: 30
Enter your choice: 1
Enter the value to append: 40
Enter your choice: 1
Enter the value to append: 50
Enter your choice: 2
Enter your choice: 3
Odd Position
10 30 50
Even Position
20 40
```

Q5:- W.A.P. to create a binary search tree and perform following operations:

1) Search a particular key.
2) Delete a node from the tree.
3) Count total number of leaf nodes
4) Count nodes having both children in the binary search tree
5) Count total numbers of nodes from right left side of root node

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

typedef struct tree
{
   int data;
   struct tree *left, *right;
} node;

node *insert(node *root, int x)
{
   if (root == NULL)
   {
      node *nn = (node *)malloc(sizeof(node));
      nn->left = nn->right = NULL;
      nn->data = x;
      return nn;
   }
   else if (root->data > x)
      root->left = insert(root->left, x);
   else if (root->data < x)
      root->right = insert(root->right, x);
   return root;
}

node *search(node *root, int key)
{
   if (root == NULL || root->data == key)
      return root;

   if (root->data > key)
      return search(root->left, key);

   return search(root->right, key);
}

node *min(node *root)
{
   if (root == NULL)
      return NULL;
   else if (root->left == NULL)
```

```
        return root;
    else
        return min(root->left);
}

node *deleteNode(node *root, int key)
{
    if (root == NULL)
        return root;

    if (key < root->data)
        root->left = deleteNode(root->left, key);
    else if (key > root->data)
        root->right = deleteNode(root->right, key);
    else
    {
        if (root->left == NULL)
        {
            node *temp = root->right;
            free(root);
            return temp;
        }
        else if (root->right == NULL)
        {
            node *temp = root->left;
            free(root);
            return temp;
        }

        node *temp = min(root->right);
        root->data = temp->data;
        root->right = deleteNode(root->right, temp->data);
    }
    return root;
}

void leafNode(node *root, int *ct)
{
    if (root != NULL)
    {
        if (root->left == NULL && root->right == NULL)
            (*ct)++;
        leafNode(root->left, ct);
        leafNode(root->right, ct);
    }
}

void twoChildNode(node *root, int *ct)
```

```
{
   if (root != NULL)
   {
      if (root->left != NULL && root->right != NULL)
         (*ct)++;
      twoChildNode(root->left, ct);
      twoChildNode(root->right, ct);
   }
}

int countlft(node *root)
{
   if (root == NULL || (root->left == NULL && root->right == NULL))
      return 0;

   return 1 + countlft(root->left) + countlft(root->right);
}

int main()
{
   node *root = NULL;
   int leaf = 0, twoChild = 0;
   int choice, n, key;
   printf("\n\nKrishant Chauhan MCA 2A Roll no-32 \n");
   printf("Choice:\n 1-Insert\n 2-Search\n 3-Delete\n 4-Total Leaf Nodes\n 5-Ct Total Nodes with 2
Children\n 6-Count Nodes on Left Side\n 7-Exit\n\n");

   do
   {
      printf("ENTER YOUR CHOICE : ");
      scanf("%d", &choice);

      switch (choice)
      {
      case 1:
      {
         printf("ENTER THE NODE VALUE: ");
         scanf("%d", &n);
         root = insert(root, n);
         break;
      }
      case 2:
      {
         printf("Enter the value to search");
         scanf("%d", &key);
         node *result = search(root, key);
         if (result != NULL)
            printf("Value %d found\n", key);
            else
                printf("Value %d not found \n", key);
            break;
        }
        case 3:
        {
            printf("Enter the value to delete: ");
            scanf("%d", &key);
            root = deleteNode(root, key);
            printf("Node with value %d deleted \n", key);
            break;
        }
        case 4:
        {
            leafNode(root, &leaf);
            printf("Total No of Leaf Nodes are: %d\n", leaf);
            leaf = 0;
            break;
        }
        case 5:
        {
            twoChildNode(root, &twoChild);
            printf("Total No of Nodes with 2 children: %d\n", twoChild);
            twoChild = 0;
            break;
        }
        case 6:
        {
            int leftCount = countlft(root);
            printf("Total No of Nodes on the Left Side %d\n", leftCount);
            break;
        }
        case 7:
        {
            printf("Exit\n");
            break;
        }
        default:
        {
            printf("Invalid choice!\n");
            break;
        }
        }
    } while (choice != 7);

    return 0;
}
```

```
Krishant Chauhan MCA 2A Roll no-32
Choice:
 1-Insert
 2-Search
 3-Delete
 4-Total Leaf Nodes
 5-Ct Total Nodes with 2 Children
 6-Count Nodes on Left Side
 7-Exit

ENTER YOUR CHOICE : 1
ENTER THE NODE VALUE: 40
ENTER YOUR CHOICE : 1
ENTER THE NODE VALUE: 15
ENTER YOUR CHOICE : 1
ENTER THE NODE VALUE: 100
ENTER YOUR CHOICE : 1
ENTER THE NODE VALUE: 60
ENTER YOUR CHOICE : 1
ENTER THE NODE VALUE: 120
ENTER YOUR CHOICE : 1
ENTER THE NODE VALUE: 10
ENTER YOUR CHOICE : 1
ENTER THE NODE VALUE: 5
ENTER YOUR CHOICE : 2
Enter the value to search100
Value 100 found
ENTER YOUR CHOICE : 4
Total No of Leaf Nodes are: 3
ENTER YOUR CHOICE : 5
Total No of Nodes with 2 children: 2
ENTER YOUR CHOICE : 6
Total No of Nodes on the Left Side 4
ENTER YOUR CHOICE : 3
Enter the value to delete: 100
Node with value 100 deleted
```

Q6. Write a program to add of two polynomials of degree n, using linked list

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

typedef struct node
{
   int cof;
   int power;
   struct node *next;
} node;

void insert(node **head, int cof, int power)
{
   node *nn = (node *)malloc(sizeof(node));
   nn->cof = cof;
   nn->power = power;
   nn->next = NULL;

   if (*head == NULL)
   {
      *head = nn;
   }
   else
   {
      node *curr = *head;
      while (curr->next != NULL)
      {
         curr = curr->next;
      }
      curr->next = nn;
   }
}

void display(node *head)
{
   node *curr = head;
   while (curr != NULL)
   {
      printf("%dx^%d", curr->cof, curr->power);
      if (curr->next != NULL)
      {
         printf(" + ");
      }
      curr = curr->next;
   }
   printf("\n");
}
```

```
node *add(node *p1, node *p2)
{
    node *result = NULL;
    node *curr = NULL;

    while (p1 != NULL && p2 != NULL)
    {
        node *nn = (node *)malloc(sizeof(node));
        nn->next = NULL;

        if (p1->power == p2->power)
        {
            nn->cof = p1->cof + p2->cof;
            nn->power = p1->power;
            p1 = p1->next;
            p2 = p2->next;
        }
        else if (p1->power > p2->power)
        {
            nn->cof = p1->cof;
            nn->power = p1->power;
            p1 = p1->next;
        }
        else
        {
            nn->cof = p2->cof;
            nn->power = p2->power;
            p2 = p2->next;
        }

        if (result == NULL)
        {
            result = nn;
            curr = result;
        }
        else
        {
            curr->next = nn;
            curr = curr->next;
        }
    }

    while (p1 != NULL)
    {
        node *nn = (node *)malloc(sizeof(node));
        nn->cof = p1->cof;
        nn->power = p1->power;
        nn->next = NULL;

        curr->next = nn;
        curr = curr->next;

        p1 = p1->next;
    }

    while (p2 != NULL)
    {
        node *nn = (node *)malloc(sizeof(node));
        nn->cof = p2->cof;
        nn->power = p2->power;
        nn->next = NULL;

        curr->next = nn;
        curr = curr->next;

        p2 = p2->next;
    }

    return result;
}

int main()
{
    node *p1 = NULL;
    node *p2 = NULL;

    int ch, cof, power;

    printf("Menu\n");
    printf("1. Insert into Equation 1\n");
    printf("2. Insert into Equation 2\n");
    printf("3. Add Equations\n");
    printf("4. Exit\n");
    do
    {
        printf("Enter your choice: ");
        scanf("%d", &ch);

        switch (ch)
        {
        case 1:
            printf("Enter coefficient and Exponent of Equation 1: ");
            scanf("%d%d", &cof, &power);
            insert(&p1, cof, power);
            break;
        case 2:
            printf("Enter coefficient and Exponent of Equation 2: ");
            scanf("%d%d", &cof, &power);
            insert(&p2, cof, power);
            break;
        case 3:
            if (p1 == NULL || p2 == NULL)
            {
                printf("Please enter Equation 1 and Equation 2.\n");
                break;
            }
            node *p3 = add(p1, p2);
            printf("\nThe Sum of Two Equations is:\n");
            display(p3);
            break;
        case 4:
            printf("Exit\n");
            break;
        default:
            printf("Invalid choice!\n");
            break;
        }
    } while (ch != 4);
    return 0;
}
```

OUTPUT :-

```
Menu
1. Insert into Equation 1
2. Insert into Equation 2
3. Add Equations
4. Exit
Enter your choice: 1
Enter coefficient and Exponent of Equation 1: 5 3
Enter your choice: 1
Enter coefficient and Exponent of Equation 1: 7 2
Enter your choice: 1
Enter coefficient and Exponent of Equation 1: 9 0
Enter your choice: 2
Enter coefficient and Exponent of Equation 2: 8 2
Enter your choice: 2
Enter coefficient and Exponent of Equation 2: 6 1
Enter your choice: 2
Enter coefficient and Exponent of Equation 2: 3 0
Enter your choice: 3

The Sum of Two Equations is:
5x^3 + 15x^2 + 6x^1 + 12x^0
```

Q7. Write a C program to sort a sequence of characters given by user in an array, using Quick sort technique.

```
#include <stdio.h>

void swap(int *a, int *b)
{
   int temp = *a;
   *a = *b;
   *b = temp;
}

int partition(int arr[], int low, int high)
{
   int pivot = arr[high];
   int i = (low - 1);
   for (int j = low; j <= high - 1; j++)
   {
      if (arr[j] < pivot)
      {
         i++;
         swap(&arr[i], &arr[j]);
      }
   }
   swap(&arr[i + 1], &arr[high]);
   return (i + 1);
}

void quickSort(int arr[], int low, int high)
{
   if (low < high)
   {
      int pi = partition(arr, low, high);
      quickSort(arr, low, pi - 1);
      quickSort(arr, pi + 1, high);
   }
}

int main()
{
   int n = 0;
   int choice;
   int arr[100];

   printf("Menu\n");
   printf("1. Enter Characters\n");
   printf("2. Sort Characters\n");
   printf("3. Exit\n");
```

```
    do
    {
        printf("\nEnter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            printf("Enter ch: ");
            char ch;
            scanf(" %c", &ch);
            arr[n++] = ch - 'a';
            break;
        case 2:
            quickSort(arr, 0, n - 1);
            printf("\n\nAfter Sorted Sequence is: ");
            for (int i = 0; i < n; i++)
                printf("%c ", arr[i] + 'a');
            break;
        case 3:
            printf("Exiting...\n");
            break;
        default:
            printf("Invalid choice! Please try again.\n");
            break;
        }
    } while (choice != 3);

    return 0;
}
```

OUTPUT :-

```
Menu
Enter Characters
Sort Characters
Exit
Enter your choice: 1
Enter ch: k
Enter your choice: 1
Enter ch: r
Enter your choice: 1
Enter ch: i
Enter your choice: 1
Enter ch: s
Enter your choice: 1
Enter ch: i
Enter your choice: 1
Enter ch: y
Enter your choice: 1
Enter ch: a
Enter your choice: 2
After Sorted Sequence is: a i i k r s y
```

Q8. Using circular linked list allocate time slots of 10ms for given processes in time sharing Environment and then print which process will be completed in how much time.

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

typedef struct Node
{
   int id, time;
   struct Node *next;
} node;

node *append()
{
   static int id = 1;
   node *temp = malloc(sizeof(node));
   printf("Burst Time: ");
   scanf("%d", &temp->time);
   temp->id = id++;
   temp->next = NULL;
   return temp;
}

node *insert(node *head)
{
   node *temp = append();
   if (head == NULL)
   {
      temp->next = temp;
      return temp;
   }

   temp->next = head->next;
   head->next = temp;
   head = temp;
   return head;
}

node *delete_node(node *head)
{
   node *curr = head;
   if (head == NULL)
      return head;

   if (head == head->next)
   {
      free(head);
      return NULL;
   }
```

```
    curr = curr->next;
    head->id = curr->id;
    head->time = curr->time;
    head->next = curr->next;
    free(curr);
    return head;
}

node *process(node *head)
{
    int quantum = 10;
    int curr_time = 0;

    while (head != NULL)
    {
        if (head->time <= quantum)
        {
            curr_time += head->time;
            printf("Process ID : %d\n", head->id);
            printf("Process Done:- %dms\n", curr_time);
            head = delete_node(head);
        }
        else
        {
            curr_time += quantum;
            head->time -= quantum;
            head = head->next;
        }
    }
    return head;
}

int main()
{
    node *head = NULL;
    int ch;
    printf("\n\nKrishant Chauhan MCA 2A Roll no-32 \n");
    printf("Menu\n");
    printf("1. Insert process\n");
    printf("2. Start process\n");
    printf("3. Exit\n");
    do
    {
        printf("Enter your choice: ");
        scanf("%d", &ch);

        switch (ch)
        {
        case 1:
            head = insert(head);
            break;
        case 2:
            if (head != NULL)
                head = process(head->next);
            break;
        case 3:
            printf("Exit\n");
            break;
        default:
            printf("Invalid choice!\n");
            break;
        }
        printf("\n");
    } while (ch != 3);

    return 0;
}
```

Output:-

```
Krishant Chauhan MCA 2A Roll no-32
Menu
Insert process
Start process
Exit

Enter your choice: 1
Burst Time: 10
Enter your choice: 1
Burst Time: 30
Enter your choice: 1
Burst Time: 50
Enter your choice: 1
Burst Time: 70
Enter your choice: 2
Process ID: 1
Process Done: - 10ms
Process ID: 2
Process Done:- 80ms
Process ID: 3
Process Done:- 130ms
Process ID: 4
Process Done: - 160ms
```

Q9. Write a C program to store the details of a weighted graph (Use array of pointers concept).

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

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

void insert(node *graph[], int src, int dst, int weight)
{
   node *nn1 = malloc(sizeof(node));
   node *nn2 = malloc(sizeof(node));

   nn1->data = dst;
   nn1->weight = weight;
   nn1->next = graph[src];
   graph[src] = nn1;

   nn2->data = src;
   nn2->weight = weight;
   nn2->next = graph[dst];
   graph[dst] = nn2;
}

void disp(node *head, int src)
{
   while (head != NULL)
   {
      printf("%d --> %d (weight: %d)\n", src, head->data, head->weight);
      head = head->next;
   }
}

int main()
{
   node *graph[max] = {NULL};
   int ch, src, dst;

   printf("\n\nKrishant Chauhan MCA 2A Roll no-32 \n");
   printf("Menu\n");
   printf("1. Insert\n");
   printf("2. Display\n");
   printf("3. Exit\n");
```

```
    do
    {
        printf("Enter your choice: ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            printf("Enter source, destination, and weight:- ");
            scanf("%d%d%d", &src, &dst, &ch);
            insert(graph, src, dst, ch);
            break;
        case 2:
            for (ch = 0; ch < max; ch++)
            {
                if (graph[ch] != NULL)
                {
                    disp(graph[ch], ch);
                    printf("\n");
                }
            }
            break;
        case 3:
            printf("Exit\n");
            break;
        default:
            printf("Invalid choice!\n");
            break;
        }
        printf("\n");
    } while (ch != 3);

    return 0;
}
```

Output:-

```
Krishant Chauhan MCA 2A Roll no-32
Menu
1. Insert
2. Display
3. Exit
Enter your choice: 1
Enter source, destination, and weight:- 0 1 5

Enter your choice: 1
Enter source, destination, and weight:- 1 2 2

Enter your choice: 1
Enter source, destination, and weight:- 1 3 4

Enter your choice: 2
0 --> 1 (weight: 5)

1 --> 3 (weight: 4)
1 --> 2 (weight: 2)
1 --> 0 (weight: 5)

2 --> 1 (weight: 2)

3 --> 1 (weight: 4)
```

Q10. Write a C program to implement BFS.

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

struct node
{
   int vertex;
   struct node *next;
};

struct Graph
{
   int numVertices;
   struct node **adjLists;
};

struct node *createnode(int v)
{
   struct node *newnode = (struct node *)malloc(sizeof(struct node));
   newnode->vertex = v;
   newnode->next = NULL;
   return newnode;
}

struct Graph *createGraph(int numVertices)
{
   struct Graph *graph = (struct Graph *)malloc(sizeof(struct Graph));
   graph->numVertices = numVertices;
   graph->adjLists = (struct node *)malloc(numVertices * sizeof(struct node));

   int i;
   for (i = 0; i < numVertices; i++)
      graph->adjLists[i] = NULL;

   return graph;
}

void addEdge(struct Graph *graph, int src, int dest)
{
   struct node *newnode = createnode(dest);
   newnode->next = graph->adjLists[src];
   graph->adjLists[src] = newnode;

   newnode = createnode(src);
   newnode->next = graph->adjLists[dest];
   graph->adjLists[dest] = newnode;
}
```

```
void printGraph(struct Graph *graph)
{
   int i;
   for (i = 0; i < graph->numVertices; i++)
   {
      struct node *temp = graph->adjLists[i];
      printf("Adjacency list of vertex %d\n", i);
      while (temp)
      {
         printf("%d -> ", temp->vertex);
         temp = temp->next;
      }
      printf("NULL\n");
   }
}

void BFS(struct Graph *graph, int startVertex)
{
   int *visited = (int *)malloc(graph->numVertices * sizeof(int));
   int i;
   for (i = 0; i < graph->numVertices; i++)
      visited[i] = 0;

   int *queue = (int *)malloc(graph->numVertices * sizeof(int));
   int front = 0, rear = 0;

   visited[startVertex] = 1;
   queue[rear++] = startVertex;

   printf("BFS traversal starting from vertex %d: ", startVertex);

   while (front < rear)
   {
      int currentVertex = queue[front++];
      printf("%d ", currentVertex);

      struct node *temp = graph->adjLists[currentVertex];

      while (temp)
      {
         int adjVertex = temp->vertex;

         if (!visited[adjVertex])
         {
            visited[adjVertex] = 1;
            queue[rear++] = adjVertex;
         }
```

```
            temp = temp->next;
        }
    }

    printf("\n");

    free(visited);
    free(queue);
}

int main()
{
    int numVertices, choice, src, dest, startVertex;

    printf("Enter the number of vertices in the graph: ");
    scanf("%d", &numVertices);

    struct Graph *graph = createGraph(numVertices);

    printf("\nMenu\n");
    printf("1. Add an edge\n");
    printf("2. Print the graph\n");
    printf("3. Perform BFS traversal\n");
    printf("4. Exit\n");
    while (1)
    {
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            printf("Enter the source and destination vertices of the edge: ");
            scanf("%d %d", &src, &dest);
            addEdge(graph, src, dest);
            break;

        case 2:
            printGraph(graph);
            break;

        case 3:
            printf("Enter the starting vertex for BFS traversal: ");
            scanf("%d", &startVertex);
            BFS(graph, startVertex);
            break;

        case 4:
            printf("Exit\n");
            exit(0);

        default:
            printf("Invalid choice!\n");
        }
    }

    return 0;
}
```

Output:-

```
Enter the number of vertices in the graph: 4

Menu
1. Add an edge
2. Print the graph
3. Perform BFS traversal
4. Exit
Enter your choice: 1
Enter the source and destination vertices of the edge: 0 1

Enter your choice: 1
Enter the source and destination vertices of the edge: 0 2

Enter your choice: 1
Enter the source and destination vertices of the edge: 1 2

Enter your choice: 1
Enter the source and destination vertices of the edge: 2 3

Enter your choice: 2
Adjacency list of vertex 0
2 -> 1 -> NULL
Adjacency list of vertex 1
2 -> 0 -> NULL
Adjacency list of vertex 2
3 -> 1 -> 0 -> NULL
Adjacency list of vertex 3
2 -> NULL

Enter your choice: 3
Enter the starting vertex for BFS traversal: 0
BFS traversal starting from vertex 0: 0 2 1 3

Enter your choice: 4
Exit
```